

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON NEXT PAGE OF THIS FORM.)

I. (a) PLAINTIFFS

EARLY WARNING SERVICES, LLC

(b) County of Residence of First Listed Plaintiff Maricopa County, AZ  
(EXCEPT IN U.S. PLAINTIFF CASES)

(c) Attorneys (Firm Name, Address, and Telephone Number)

Daniel J. Goettle, Baker & Hostetler LLP, Cira Centre, 12th Fl.,  
2929 Arch St., Philadelphia, PA 19104, 215-568-3100

DEFENDANTS

WILLIAM GRECIA,

County of Residence of First Listed Defendant Chester County  
(IN U.S. PLAINTIFF CASES ONLY)

NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF  
THE TRACT OF LAND INVOLVED.

Attorneys (If Known)

Matthew Wawryn, Wawrzyn LLC, 200 East Randolph St.,  
Suite 5100, Chicago, IL 60601, 312-235-3120

II. BASIS OF JURISDICTION (Place an "X" in One Box Only)

- ☐ 1 U.S. Government Plaintiff ☒ 3 Federal Question (U.S. Government Not a Party)
- ☐ 2 U.S. Government Defendant ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)

III. CITIZENSHIP OF PRINCIPAL PARTIES (Place an "X" in One Box for Plaintiff and One Box for Defendant)

- |   | PTF                        | DEF                        |   | PTF                        | DEF                        |
|---|----------------------------|----------------------------|---|----------------------------|----------------------------|
| Citizen of This State                   | <input type="checkbox"/> 1 | <input type="checkbox"/> 1 | Incorporated or Principal Place of Business In This State     | <input type="checkbox"/> 4 | <input type="checkbox"/> 4 |
| Citizen of Another State                | <input type="checkbox"/> 2 | <input type="checkbox"/> 2 | Incorporated and Principal Place of Business In Another State | <input type="checkbox"/> 5 | <input type="checkbox"/> 5 |
| Citizen or Subject of a Foreign Country | <input type="checkbox"/> 3 | <input type="checkbox"/> 3 | Foreign Nation  | <input type="checkbox"/> 6 | <input type="checkbox"/> 6 |

IV. NATURE OF SUIT (Place an "X" in One Box Only)

Click here for: [Nature of Suit Code Descriptions.](#)

CONTRACT	TORTS	FORFEITURE/PENALTY	BANKRUPTCY	OTHER STATUTES
<input type="checkbox"/> 110 Insurance <input type="checkbox"/> 120 Marine <input type="checkbox"/> 130 Miller Act <input type="checkbox"/> 140 Negotiable Instrument <input type="checkbox"/> 150 Recovery of Overpayment & Enforcement of Judgment <input type="checkbox"/> 151 Medicare Act <input type="checkbox"/> 152 Recovery of Defaulted Student Loans (Excludes Veterans) <input type="checkbox"/> 153 Recovery of Overpayment of Veteran's Benefits <input type="checkbox"/> 160 Stockholders' Suits <input type="checkbox"/> 190 Other Contract <input type="checkbox"/> 195 Contract Product Liability <input type="checkbox"/> 196 Franchise	<b>PERSONAL INJURY</b> <input type="checkbox"/> 310 Airplane <input type="checkbox"/> 315 Airplane Product Liability <input type="checkbox"/> 320 Assault, Libel & Slander <input type="checkbox"/> 330 Federal Employers' Liability <input type="checkbox"/> 340 Marine <input type="checkbox"/> 345 Marine Product Liability <input type="checkbox"/> 350 Motor Vehicle <input type="checkbox"/> 355 Motor Vehicle Product Liability <input type="checkbox"/> 360 Other Personal Injury <input type="checkbox"/> 362 Personal Injury - Medical Malpractice	<input type="checkbox"/> 365 Personal Injury - Product Liability <input type="checkbox"/> 367 Health Care/Pharmaceutical Personal Injury Product Liability <input type="checkbox"/> 368 Asbestos Personal Injury Product Liability <b>LABOR</b> <input type="checkbox"/> 370 Other Fraud <input type="checkbox"/> 371 Truth in Lending <input type="checkbox"/> 380 Other Personal Property Damage <input type="checkbox"/> 385 Property Damage Product Liability	<input type="checkbox"/> 422 Appeal 28 USC 158 <input type="checkbox"/> 423 Withdrawal 28 USC 157 <b>PROPERTY RIGHTS</b> <input type="checkbox"/> 820 Copyrights <input checked="" type="checkbox"/> 830 Patent <input type="checkbox"/> 835 Patent - Abbreviated New Drug Application <input type="checkbox"/> 840 Trademark <input type="checkbox"/> 880 Defend Trade Secrets Act of 2016 <b>SOCIAL SECURITY</b> <input type="checkbox"/> 861 HIA (1395ff) <input type="checkbox"/> 862 Black Lung (923) <input type="checkbox"/> 863 DIWC/DIWW (405(g)) <input type="checkbox"/> 864 SSID Title XVI <input type="checkbox"/> 865 RSI (405(g)) <b>FEDERAL TAX SUITS</b> <input type="checkbox"/> 870 Taxes (U.S. Plaintiff or Defendant) <input type="checkbox"/> 871 IRS—Third Party 26 USC 7609	<input type="checkbox"/> 375 False Claims Act <input type="checkbox"/> 376 Qui Tam (31 USC 3729(a)) <input type="checkbox"/> 400 State Reapportionment <input type="checkbox"/> 410 Antitrust <input type="checkbox"/> 430 Banks and Banking <input type="checkbox"/> 450 Commerce <input type="checkbox"/> 460 Deportation <input type="checkbox"/> 470 Racketeer Influenced and Corrupt Organizations <input type="checkbox"/> 480 Consumer Credit (15 USC 1681 or 1692) <input type="checkbox"/> 485 Telephone Consumer Protection Act <input type="checkbox"/> 490 Cable/Sat TV <input type="checkbox"/> 850 Securities/Commodities/Exchange <input type="checkbox"/> 890 Other Statutory Actions <input type="checkbox"/> 891 Agricultural Acts <input type="checkbox"/> 893 Environmental Matters <input type="checkbox"/> 895 Freedom of Information Act <input type="checkbox"/> 896 Arbitration <input type="checkbox"/> 899 Administrative Procedure Act/Review or Appeal of Agency Decision <input type="checkbox"/> 950 Constitutionality of State Statutes
<b>REAL PROPERTY</b> <input type="checkbox"/> 210 Land Condemnation <input type="checkbox"/> 220 Foreclosure <input type="checkbox"/> 230 Rent Lease & Ejectment <input type="checkbox"/> 240 Torts to Land <input type="checkbox"/> 245 Tort Product Liability <input type="checkbox"/> 290 All Other Real Property	<b>CIVIL RIGHTS</b> <input type="checkbox"/> 440 Other Civil Rights <input type="checkbox"/> 441 Voting <input type="checkbox"/> 442 Employment <input type="checkbox"/> 443 Housing/Accommodations <input type="checkbox"/> 445 Amer. w/Disabilities - Employment <input type="checkbox"/> 446 Amer. w/Disabilities - Other <input type="checkbox"/> 448 Education	<b>PRISONER PETITIONS</b> <b>Habeas Corpus:</b> <input type="checkbox"/> 463 Alien Detainee <input type="checkbox"/> 510 Motions to Vacate Sentence <input type="checkbox"/> 530 General <input type="checkbox"/> 535 Death Penalty <b>Other:</b> <input type="checkbox"/> 540 Mandamus & Other <input type="checkbox"/> 550 Civil Rights <input type="checkbox"/> 555 Prison Condition <input type="checkbox"/> 560 Civil Detainee - Conditions of Confinement	<input type="checkbox"/> 625 Drug Related Seizure of Property 21 USC 881 <input type="checkbox"/> 690 Other <input type="checkbox"/> 710 Fair Labor Standards Act <input type="checkbox"/> 720 Labor/Management Relations <input type="checkbox"/> 740 Railway Labor Act <input type="checkbox"/> 751 Family and Medical Leave Act <input type="checkbox"/> 790 Other Labor Litigation <input type="checkbox"/> 791 Employee Retirement Income Security Act <b>IMMIGRATION</b> <input type="checkbox"/> 462 Naturalization Application <input type="checkbox"/> 465 Other Immigration Actions	

V. ORIGIN (Place an "X" in One Box Only)

- ☒ 1 Original Proceeding ☐ 2 Removed from State Court ☐ 3 Remanded from Appellate Court ☐ 4 Reinstated or Reopened ☐ 5 Transferred from Another District (specify) ☐ 6 Multidistrict Litigation - Transfer ☐ 8 Multidistrict Litigation - Direct File

VI. CAUSE OF ACTION

Cite the U.S. Civil Statute under which you are filing (Do not cite jurisdictional statutes unless diversity):  
35 U.S.C. § 1 et seq., 28 U.S.C. §§ 2201 and 2202

Brief description of cause:

3. This is a civil action for declaratory judgment of invalidity and non-infringement of a patent.

VII. REQUESTED IN COMPLAINT:

☐ CHECK IF THIS IS A CLASS ACTION UNDER RULE 23, F.R.Cv.P. DEMAND \$ CHECK YES only if demanded in complaint:  
JURY DEMAND: ☐ Yes ☐ No

VIII. RELATED CASE(S) IF ANY

(See instructions):

JUDGE \_\_\_\_\_ DOCKET NUMBER \_\_\_\_\_

DATE 03/04/2021 SIGNATURE OF ATTORNEY OF RECORD /s/ Daniel J. Goettle

FOR OFFICE USE ONLY

RECEIPT # \_\_\_\_\_ AMOUNT \_\_\_\_\_ APPLYING IFP \_\_\_\_\_ JUDGE \_\_\_\_\_ MAG. JUDGE \_\_\_\_\_

**DESIGNATION FORM**

(to be used by counsel or pro se plaintiff to indicate the category of the case for the purpose of assignment to the appropriate calendar)

Address of Plaintiff: Early Warning Services, LLC, 16552 N. 90th St., Scottsdale, AZ 85260

Address of Defendant: William Grecia, 121 Lenora Ln, Downingtown, PA, 19335-1142

Place of Accident, Incident or Transaction: Phoenix, Arizona

**RELATED CASE, IF ANY:**

Case Number: \_\_\_\_\_ Judge: \_\_\_\_\_ Date Terminated: \_\_\_\_\_

Civil cases are deemed related when **Yes** is answered to any of the following questions:

- |  |                              |  |
|--|------------------------------|--|
| 1. Is this case related to property included in an earlier numbered suit pending or within one year previously terminated action in this court?  | Yes <input type="checkbox"/> | No <input checked="" type="checkbox"/> |
| 2. Does this case involve the same issue of fact or grow out of the same transaction as a prior suit pending or within one year previously terminated action in this court?            | Yes <input type="checkbox"/> | No <input checked="" type="checkbox"/> |
| 3. Does this case involve the validity or infringement of a patent already in suit or any earlier numbered case pending or within one year previously terminated action of this court? | Yes <input type="checkbox"/> | No <input checked="" type="checkbox"/> |
| 4. Is this case a second or successive habeas corpus, social security appeal, or pro se civil rights case filed by the same individual?  | Yes <input type="checkbox"/> | No <input checked="" type="checkbox"/> |

I certify that, to my knowledge, the within case ☐ is / ☒ is not related to any case now pending or within one year previously terminated action in this court except as noted above.

DATE: 03/04/2021 /s/ Daniel J. Goettle 85046  
*Must sign here*  
Attorney-at-Law / Pro Se Plaintiff Attorney I.D. # (if applicable)

**CIVIL: (Place a ✓ in one category only)**

**A. Federal Question Cases:**

- |                                     |   |
|-------------------------------------|---|
| <input type="checkbox"/>            | 1. Indemnity Contract, Marine Contract, and All Other Contracts |
| <input type="checkbox"/>            | 2. FELA   |
| <input type="checkbox"/>            | 3. Jones Act-Personal Injury                                    |
| <input type="checkbox"/>            | 4. Antitrust  |
| <input checked="" type="checkbox"/> | 5. Patent   |
| <input type="checkbox"/>            | 6. Labor-Management Relations                                   |
| <input type="checkbox"/>            | 7. Civil Rights   |
| <input type="checkbox"/>            | 8. Habeas Corpus  |
| <input type="checkbox"/>            | 9. Securities Act(s) Cases                                      |
| <input type="checkbox"/>            | 10. Social Security Review Cases                                |
| <input type="checkbox"/>            | 11. All other Federal Question Cases                            |
- (Please specify): \_\_\_\_\_

**B. Diversity Jurisdiction Cases:**

- |                          |  |
|--------------------------|--|
| <input type="checkbox"/> | 1. Insurance Contract and Other Contracts        |
| <input type="checkbox"/> | 2. Airplane Personal Injury                      |
| <input type="checkbox"/> | 3. Assault, Defamation                           |
| <input type="checkbox"/> | 4. Marine Personal Injury                        |
| <input type="checkbox"/> | 5. Motor Vehicle Personal Injury                 |
| <input type="checkbox"/> | 6. Other Personal Injury (Please specify): _____ |
| <input type="checkbox"/> | 7. Products Liability                            |
| <input type="checkbox"/> | 8. Products Liability – Asbestos                 |
| <input type="checkbox"/> | 9. All other Diversity Cases                     |
- (Please specify): \_\_\_\_\_

**ARBITRATION CERTIFICATION**

(The effect of this certification is to remove the case from eligibility for arbitration.)

I, Daniel J. Goettle, counsel of record or pro se plaintiff, do hereby certify:

☐ Pursuant to Local Civil Rule 53.2, § 3(c) (2), that to the best of my knowledge and belief, the damages recoverable in this civil action case exceed the sum of \$150,000.00 exclusive of interest and costs:

☒ Relief other than monetary damages is sought.

DATE: 03/04/2021 /s/ Daniel J. Goettle 85046  
*Sign here if applicable*  
Attorney-at-Law / Pro Se Plaintiff Attorney I.D. # (if applicable)

NOTE: A trial de novo will be a trial by jury only if there has been compliance with F.R.C.P. 38.

IN THE UNITED STATES DISTRICT COURT  
FOR THE EASTERN DISTRICT OF PENNSYLVANIA

EARLY WARNING SERVICES, LLC

PLAINTIFF,

v.

WILLIAM GRECIA,

DEFENDANT.

Case No. \_\_\_\_\_

**COMPLAINT FOR DECLARATORY JUDGMENT OF PATENT INVALIDITY AND  
NON-INFRINGEMENT**

Plaintiff Early Warning Services, LLC (“EWS”) alleges as follows against defendant William Grecia (“Grecia”).

**I. PARTIES**

1. EWS is a Delaware limited liability company having a principal place of business at 16552 N. 90th St., Scottsdale, AZ 85260.

2. Grecia is an individual who, upon information and belief, is domiciled in Downingtown, Pennsylvania.

**II. NATURE OF THE ACTION**

3. This is a civil action for declaratory judgment of invalidity and non-infringement of Grecia’s patent under the Declaratory Judgement Act, 28 U.S.C. §§ 2201 and 2202.

**III. JURISDICTION AND VENUE**

4. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a), the Patent Laws of the United States, 35 U.S.C. §§ 1 *et seq.*, particularly 35 U.S.C §§ 101, 102, 103, 112, and/or 271, and the Declaratory Judgment Act, 28 U.S.C. §§ 2201 and 2202.

5. The Court has personal jurisdiction over Grecia as he is an individual residing in this District. Grecia identified this District as his residence in Application Disclosure Statements (“ADS”) filed with the United States Patent & Trademark Office (“USPTO” or “Patent Office”). For example, on July 31, 2014, Grecia signed and submitted an ADS in connection with U.S. Patent No. 8,887,308 wherein he identified his residence as Downingtown, PA. *See* Exhibit 1. Moreover, Grecia’s publicly available LinkedIn’s webpage identifies his location as Downingtown, PA, and Grecia has alleged in Federal Court complaints filed as recently as February 6, 2021 that he is “residing in Downingtown, Pennsylvania.” *See* Exhibit 2.

6. Upon information and belief, Grecia regularly transacts business in this judicial district and elsewhere in the United States, in part, by enforcing his patent rights against third parties through licenses or litigation involving his patent portfolio, including U.S. Patent No. 8,402,555 (“the ’555 Patent”). For example, Grecia filed over 50 federal lawsuits for purported infringement of his patents against numerous corporate defendants such as Google Inc., Apple Inc., Sony Network Entertainment International, LLC, Samsung Electronics America, Inc., Microsoft Corp., and Amazon.com. The past corporate targets of Grecia’s frequent patent infringement suits include four of EWS’ customers, namely, The Bank of New York Mellon Corporation, Citibank, N.A., Morgan Stanley Smith Barney, LLC, and TIAA, FSB d/b/a TIAA Bank. In the prior lawsuits against these four EWS customers, the court invalidated a different one of Grecia’s patents, and Grecia did not appeal the court’s order.

7. Grecia continues to attempt to contort the scope of his patents in order to pursue claims against participants in the financial services industry. On December 23, 2020, at 4:04 a.m., Grecia directed his attorney to send “William Grecia’s demand” to counsel for EWS. *See* Exhibit 3.



8. Upon information and belief, Grecia enforces his patent portfolio from this District, conducts and solicits business in this District, including sending infringement demands from this District, and conducts a significant volume of business in this District.

9. Venue is proper in this judicial district pursuant to 28 U.S.C § 1391 because Grecia resides in this District and is subject to the Court’s personal jurisdiction and a substantial part of the events or omission giving rise to these claims occurred in this District.

#### **IV. FACTUAL BACKGROUND**

##### **Grecia Threatens to Sue EWS and/or its Customers**

10. EWS is the owner of the Zelle® network, a financial services network focused on transforming digital payment experiences. EWS provides certain services to its customers to enable such customers to participate in the Zelle® network. In particular, EWS’s customers in the financial industry participate in the Zelle® network and provide certain financial payment services to their respective customers.

11. On December 23, 2020, Grecia sent a “demand” letter to EWS’s counsel. In the letter, Grecia claimed enforceable rights in, to, and under the ’555 Patent, and specifically accused four (4) EWS customers of directly infringing his patent rights. Specifically, Grecia accuses EWS’s customers First National Bank of Central Texas (FNBCT), Frost Bank, American Bank, and First National Bank of Texas (FNBT) of infringing the ’555 Patent. *See* Exhibit 3.

12. Grecia contends he is the owner of the ’555 Patent, entitled “Personalized digital media access system (PDMAS).” *See* attached Exhibit 4.

13. Grecia contends he has the power and authority to enforce the ’555 Patent. *See* Exhibit 4.

14. Grecia contends he is the owner of the '555 Patent, entitled "Personalized digital media access system (PDMAS)." *See Id.*

15. Grecia contends he has the power and authority to enforce the '555 Patent. *Id.*

16. Grecia demanded EWS "move wisely" to "buy out" these cases for an "early discounted amount" before filing cases against each of EWS's customers "directly after January 5th 2021." *Id.* By doing so, Grecia accused EWS of indirectly infringing his patent rights.

17. Grecia threatens to extort larger sums of money from EWS and/or EWS's customers after filing suits against them in the Western District of Texas. *See Id.* With his demand, Grecia sent four claim charts purporting to identify EWS's customers' alleged infringement of at least claim 23 of the '555 Patent. *Id.*

18. However, Grecia's demand is not limited to the four identified EWS customers. Grecia threatens further enforcement efforts and, indicative of his bad faith, states that he will "not consider any bulk or global" deals concerning EWS's Zelle® network "at all in 2021." *Id.*

19. On January 8, 2021, Grecia filed a complaint against Frost Bank in the United States District Court for the Western District of Texas, case no. 6:21-cv-00016-ADA ("Frost Lawsuit"). *See Exhibit 5.*

20. On January 19, 2021, Frost Bank sent a letter to EWS demanding indemnification from EWS for the Frost Lawsuit filed by Grecia, and EWS has agreed to defend and indemnify Frost Bank from Grecia's allegations in the Frost Lawsuit.

21. Grecia also has indicated that he "may" in the future add EWS as a party to the Frost Lawsuit but has not taken any steps to do so.

22. In an email dated February 4, 2021, Grecia also stated that EWS' customer, First National Bank of Central Texas (FNBCT), "has a pending Zelle action" although Grecia has not yet commenced litigation against FNBCT. *See* Exhibit 6.

**The '555 Patent Recites Patent-Ineligible Subject Matter**

23. The claims of Grecia's '555 Patent are directed to the abstract idea of monitoring and protecting access to digital data to prevent unauthorized access.

24. The '555 Patent relates to the field of digital rights management ("DRM"). Exhibit 4 at 1:19–26. DRM is a generic term for algorithms used to prevent unauthorized copying or distribution of digital media, such as music, movies, or games, by restricting access to the media across devices, such as computers. *Id.* at 1:28–34.

25. According to the '555 Patent, traditional DRM schemes have two shortcomings. *Id.* at 2:54–3:2. First, traditional systems rely on digital content providers to maintain computer servers to receive and send authorization keys. *Id.* at 2:54–56. But sometimes content providers discontinue servers or go out of business after the DRM content has been sold to consumers, who then lose the ability to access content they had purchased. *Id.* at 2:59–62. Second, traditional systems often function by linking rights to access content to the machine on which the content was acquired, which means that users can lose access to the content if the machine breaks *Id.* at 2:64–3:2.

26. Grecia's DRM scheme, as described in the '555 Patent, allegedly solves these problems by "branding" the digital content with information that associates a user with that digital content. *Id.* at 3:2–7, 4:2–9. The branding takes place by writing the information into "metadata," which is a log that travels with the digital content at all times. This branding purportedly enables the user to access the digital file from any device. *Id.*

27. To implement this abstract idea of branding data with the user's permission to access that data, claim 15 simply recites a set of generic computing functions—(1) *receiving* an encrypted digital media access branding request. . . the request comprising a membership verification token. . .; (2) *authenticating* the membership verification token. . .; (3) *establishing* a connection with at least one communications console. . .; (4) *requesting* at least one identification reference. . .; (5) *receiving* at least one electronic identification reference from the at least one communications console; and (6) *branding* metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata. All of these functions are performed by generic computing devices (*e.g.*, an “apparatus”) without any inventive step.

28. Dependent claim 23, which depends from independent claim 15 simply further recites “wherein the encrypted digital media is *associated* with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.”

29. The claims of the '555 Patent, and claims 15 and 23 in particular, do not teach a specific way to improve the functionality of a computer.

30. The claims of the '555 Patent, and claims 15 and 23 in particular, are not limited to a specific technical solution in digital rights management.

31. The claims of the '555 Patent, and claims 15 and 23 in particular, recite generic computer components to control access to digital content through a series of functionally-oriented steps that employ conventional computer components.

### **Prior Art Invalidates the Claims of the '555 Patent**

32. U.S. Patent Application Publication No. 2007/0156726 to Levy, published July 5, 2007, is prior art to the '555 Patent under 35 U.S.C. § 102(b) (pre-AIA). *See* Exhibit 7.

33. Assertions and Protocols for the Security Asserting Markup Language V2.0 (“SAML”), published March 15, 2005, is prior art to the ’555 Patent under 35 U.S.C. § 102(b) (pre-AIA). *See* Exhibit 8.

34. EWS is informed and believes, and thereupon alleges that SAML was publicly available as of March 17, 2005 from OASIS (“a non-profit consortium that drives the development, convergence and adoption of open standards for the global information society”) through the OASIS website having the following URL: <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>, as confirmed by the affidavits of Messrs. Scott McGrath and Christopher Butler, *see* Exhibits 9 and 10.

35. Applied Cryptography (“Applied Cryptography”), published in 1996, is prior art to the ’555 Patent under 35 U.S.C. § 102(b) (pre-AIA). *See* Exhibit 11. EWS is informed and believes, and thereupon alleges that Applied Cryptography has been publicly available at the Cornell University Library since at least November 21, 2001.

36. The claims of the ’555 Patent, and claim 23 in particular, are invalid for being anticipated by known prior art references under 35 U.S.C. § 102 (pre-AIA), including the Levy reference.

37. The claims of the ’555 Patent, are rendered obvious by a combination of prior art references under 35 U.S.C. § 103 (pre-AIA), including the Levy, SAML, and Applied Cryptography references.

38. Each of the claims of the ’555 Patent that Grecia has asserted against EWS and its customers are invalid and/or unpatentable under, at least, 35 U.S.C. §§ 102 and 103 (pre-AIA).

**The Claims of the ’555 Patent are Indefinite**

39. The claims of the '555 Patent, and specifically at least claims 15 and 23, invoke 35 U.S.C. 112, ¶ 6 (pre-AIA) and are indefinite under 35 U.S.C. § 112, ¶ 2 (pre-AIA).

40. The “computer program product” term of claims 15 and 23 are means-plus-function terms. The computer program product of claims 15 includes “a non-transitory computer usable medium” and a “computer readable program code stored therein.” The “computer program product” phrase is expressed as means for “performing” specified function. The generic components recited by claim 15 fail to provide sufficient structure or an algorithm for performing the recited functions. For example, the '555 Patent specification does not disclose the structure that performs the functions of “receiving”, “authenticating”, “establishing”, “requesting”, “receiving”, or “branding”. Because the '555 Patent specification does not contain a disclosure of structures for performing the recited functions, claims 15 and 23 are indefinite under 35 U.S.C. § 112, ¶ 2 (pre-AIA).

41. The claims of the '555 Patent, and specifically at least claims 15 and 23, are indefinite under 35 U.S.C. § 112, ¶ 2 (pre-AIA).

42. The term “metadata of the encrypted digital media” in claim 15 is indefinite under 35 U.S.C. § 112, ¶ 2 (pre-AIA). The '555 Patent specification provides an express definition for “metadata” — “[t]o understand metadata and the uses, metadata is defined simply as to ‘describe other data’. It provides information about certain item’s content.” Exhibit 4 at 13:24-27. However, at least claim 15’s recitation of “metadata” is inconsistent with the '555 Patent specification’s express definition of “metadata.” Because one of ordinary skill in the art could not ascertain the scope of at least claims 15 and 23 with reasonable certainty, at least claims 15 and 23 are invalid.

43. The term “communication console” in claim 15 is indefinite under 35 U.S.C. § 112, ¶ 2 (pre-AIA). In the claim chart that Grecia attached to his demand letter, the “at least one communications console” does not align to a single system that is different from the system that is allegedly receiving an encrypted digital media access branding request. *See* Exhibit 4. Because one of ordinary skill in the art could not ascertain the scope of at least claims 15 and 23 with reasonable certainty, at least claims 15 and 23 are invalid.

44. Another court has also found Claims 12-14 and 24-26 of the ’555 patent indefinite and thus invalid. *See* Exhibit 12.

#### **The Claims of the ’555 Patent Lack Written Description and Are Not Enabled**

45. The claims of the ’555 Patent, and specifically at least claims 15 and 23, lack written description in and/or are not enabled by the specification and are therefore invalid under 35 U.S.C. § 112, ¶ 1 (pre-AIA).

46. For example, the following limitations of claim 15 are not supported or enabled by the specification:

- “verified web service capable of facilitating a two way data exchange to complete a verification process.”
- “facilitating interoperability between a plurality of data processing devices.”
- “the communications console is a combination of a graphic user interface (GUI) and an Application Programmable Interface (API) protocol wherein the API is related to a verified web service.”
- “metadata of the encrypted digital media” to the extent it is not limited to metadata stored within files comprising the encrypted digital content.



**Grecia Defrauds the USPTO into Issuing a Certificate of Correction That Impermissibly Broadens the Scope of the '555 Patent Claims**

47. Grecia is named as an inventor on a number of U.S. Patents. Grecia is very familiar with the prosecution process and procedure. Moreover, Grecia is well aware of his duty of candor and good faith dealing with the USPTO, which includes his duty to disclose information material to patentability. In particular, Grecia executed a Declaration for Utility or Design Patent Application (37 C.F.R. 1.63) that was filed with the USPTO on February 15, 2012 in connection with the Serial No. 13/397,517, which issued as the '555 Patent. The declaration specifically states, "I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56 . . . ." *See* Exhibit 13.

48. Grecia committed fraud on the USPTO during prosecution of the '555 Patent.

49. Grecia submitted multiple Requests for Certificates of Correction after issuance of the '555 Patent. First, in a Request for Certificate of Correction under 37 C.F.R. 1.323 submitted to the USPTO on August 16, 2013, Grecia falsely stated that the corrections were "clerical, typographical, and of minor character." *See* Exhibit 14. Grecia knew that the corrections sought were not merely clerical in nature, and he also knew that he did in fact change the scope of the claims to require additional examination. Specifically, Grecia submitted a Request for Certificate of Correction for independent claims 1, 12, and 15 of the '555 Patent and all claims dependent therefrom by changing the phrase "obtained from" to "*related to*" (emphasis added), which materially broadened the scope of the issued claims.

50. Second, in a Request for Certificate of Correction under 37 C.F.R. 1.323 submitted to the USPTO a month later on September 16, 2013, Grecia falsely stated that the corrections were of a clerical nature, of a typographical nature, or a mistake of minor character, and that the request "does not involve changes that would constitute new matter or require re-examination." *See*

Exhibit 15. Grecia knew that the corrections sought were not merely clerical in nature and did in fact change the scope of the claims to require additional examination. Specifically, Grecia submitted a Request for Certificate of Correction for dependent claims 11, 16, and 25 and all claims dependent therefrom by changing the phrase “selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by” to “selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein *the purchase permission, rental permission, and membership permission* is represented by” (emphasis added), which materially broadened the scope of the issued claims.

51. A reasonable USPTO Patent Examiner would have considered these changes material and important in deciding whether to allow the patent application. Moreover, had Grecia sought to timely amend his claims with his proposed amendments, the record would include the USPTO Patent Examiner’s search strategy and treatment of such proposed amendments. Grecia knowingly made the false statements described above with the intent to deceive the USPTO, and deprived the USPTO Patent Examiner of an opportunity to substantively review the changes by submitting them in a Request for Certificate of Correction.

52. Even after attempting to enforce the ’555 Patent against numerous other parties, Grecia, on December 23, 2020 — the day he sent his demand letter to EWS to specifically allege infringement of Claim 23 — filed a Request for Certificate of Correction regarding claim 23. *See* Exhibit 16.

53. Grecia’s deliberate false representations to the Patent Office regarding the nature and effect of his Requests for Certificates of Correction were made with the intent to deceive the Patent Office and constitute inequitable conduct.

54. But for Grecia's affirmative false representation, the claims of the '555 Patent would not have issued as amended by the Certificates of Correction (*see* Exhibits 17 and 18) because the changes sought in the Certificates of Correction were not disclosed to the USPTO during substantive prosecution of the application, which deprived the USPTO Patent Examiner of time for consideration of the scope of the terms "related to" and "selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the purchase permission, rental permission, and membership permission is represented by," and were not properly considered by the USPTO Patent Examiner.

55. Grecia's deliberate false representations to the Patent Office have injured and are injuring EWS and its customers because Grecia is now attempting to enforce unpatentable claims of the '555 Patent against EWS and its customers.

#### **Courts Have Invalidated Patents Related to the '555 Patent**

56. As alleged above, the claims of Grecia's '555 Patent are directed to the abstract idea of monitoring and protecting access to digital data to prevent unauthorized access.

57. Grecia contends he is also the owner of U.S. Patent No. 8,887,308 ("the '308 Patent"), entitled "Digital cloud access (PDMAS part III)." *See* Exhibit 19. The '308 Patent is related and claims priority to the '555 Patent. The '555 Patent and the '308 Patent share substantially identical specifications. Grecia filed a Terminal Disclaimer disclaiming the term of the '308 Patent that would extend beyond the expiration of the '555 Patent and U.S. Patent No. 8,533,860 ("the '860 Patent"). Accordingly, Grecia has admitted that the claimed invention of the '555 Patent is the same as the claimed invention of the '308 Patent.

58. Grecia asserted the sole claim of the '308 Patent in five nearly identical lawsuits against EWS's customers in the Southern District of New York. In granting EWS's customer's

motions to dismiss, the Court found the sole claim of the '308 Patent to be invalid as being directed to an abstract idea. *See* Exhibit 20, *Grecia v. Bank of New York Mellon Corp.*, 19-cv-02810-VEC, Amended Opinion and Order, Dkt. 55 (S.D.N.Y. Apr. 20, 2020), and related cases *Grecia v. CitiBank, N.A.*, 19-cv-02811, *Grecia v. Morgan Stanley Smith Barney LLC*, 19-cv-02812, *Grecia v. TIAA, FSB d/b/a TIAA Bank*, 19-cv-02813, and *Grecia v. Samsung Electronics America, Inc.*, 19-cv-03278. The claim of the '308 Patent is, for purposes of subject-matter eligibility, legally indistinguishable from the '555 Patent's claims, as evidenced by the aforementioned Terminal Disclaimer.

59. Grecia contends he is also the owner of the '860 Patent, entitled "Personalized digital media access system—PDMAS part II." *See* Exhibit 21. The '860 Patent is related and claims priority to the '555 Patent. The '555 Patent and the '860 Patent share substantially identical specifications. Grecia filed a Terminal Disclaimer disclaiming the term of the '860 Patent that would extend beyond the expiration of the '555 Patent. Accordingly, Grecia has admitted that the claimed invention of the '555 Patent is the same as the claimed invention of the '860 Patent.

60. The Southern District of New York found claims 12-14 and 24-26 of the '555 Patent and Claims 9, 10, and 21-30 of the '860 Patent invalid for indefiniteness and entered a Stipulation and Joint Motion for Entry of Final Judgement Based on the Court's Indefiniteness Rulings. *See* Exhibits 12 and 22, *Grecia v. Samsung Electronics America, Inc.*, No., 1:16-cv-09691-RJS, Dkt. 58 and 60 (S.D.N.Y. September 7 and 22, 2018).

61. On appeal, the Court of Appeals for the Federal Circuit affirmed the Southern District of New York's determination that claim 21 of the '860 Patent and claims 22, 24, 25, and 27-30 which depend therefrom are invalid as indefinite for claiming a means-plus-function term

without sufficient structure under 35 U.S.C. § 112, ¶¶ 2 and 6 (pre-AIA). *See* Exhibit 23, *Grecia v. Samsung Electronics America, Inc.*, 19-01019, Dkt. 21 (Fed. Cir. Aug. 20, 2019).

62. In an *Inter Partes* Review proceeding, the USPTO entered Judgment against Grecia and cancelled claims 1-8 and 11-20 of the '860 Patent. *Mastercard International Incorporated v. Grecia*, IPR2017-00791, Paper 13 (PTAB Sept. 27, 2017). *See* Exhibit 24.

63. Accordingly, all of the claims of the '860 Patent have been found invalid or cancelled by a Court or the USPTO.

**Grecia's Infringement Allegations Have Injured, and Are Injuring, EWS**

64. Through his demand letter, Grecia charges EWS and its customers with direct and indirect infringement of certain claims of the '555 Patent, and has created an actual case or controversy between Grecia and EWS by threatening actual and imminent injury to EWS that can be redressed by judicial relief and that injury is of sufficient immediacy and reality to warrant the issuance of a declaratory judgment.

65. Through his demand letter, Grecia further attempts to enforce against EWS and its customers claims of the '555 Patent that are unenforceable because of Grecia's deliberate false representations to the Patent Office.

66. For example, as alleged above, there is a controversy between Grecia and EWS concerning EWS's alleged liability for at least inducement of infringement based on the alleged acts of direct infringement by EWS's customers First National Bank of Central Texas (FNBCT), Frost Bank, American Bank, and First National Bank of Texas (FNBT). Accordingly, Grecia has affirmatively put EWS in a position where it must either pursue allegedly infringing behavior or abandon that which EWS has a right to do.

67. Furthermore, in response to a demand from Frost Bank, EWS has agreed to indemnify Frost Bank from Grecia's infringement allegations and is assuming the substantial cost of defending the Frost Lawsuit.

68. EWS's injury also includes uncertainty in the marketplace as to whether EWS customers' participation in the Zelle® network, including but not limited to FNBCT, Frost Bank, American Bank, and FNBT, are free from infringement based on the alleged infringement the '555 Patent's claims.

69. Absent a declaration of non-infringement, invalidity, or unenforceability of the '555 Patent, Grecia's continued wrongful assertions of infringement will cause EWS and its customers irreparable injury and damage.

70. Grecia and EWS have adverse legal interests, and there is a definite and concrete dispute between Grecia and EWS that is real, substantial, and on-going.

**EWS and its Customers Do Not Infringe the Claims of the '555 Patent**

71. EWS has not infringed, and is not infringing, either directly, indirectly, wilfully or otherwise, upon any claim, much less any valid and enforceable claim, of the '555 Patent.

72. EWS's customers, by their purchase and/or use of Zelle® network services have not infringed, and are not infringing, either directly, indirectly, wilfully or otherwise, upon any claim, much less any valid and enforceable claim, of the '555 Patent.

73. Grecia knows, and/or should have known, that EWS has not infringed, and is not infringing, either directly, indirectly wilfully or otherwise upon any claims of the '555 Patent.

74. Grecia knows, and/or should have known, that EWS's customers have not infringed, and are not infringing, either directly, indirectly wilfully or otherwise upon any claims of the '555 Patent.

75. EWS has not and is not infringing upon any claims of the '555 Patent because, among other reasons: (i) the claims of the '555 Patent are invalid and/or unenforceable; (ii) EWS does not induce any customer to infringe (or contributorily infringing) any claim of the '555 Patent; and (iii) EWS's Zelle® network does not meet all of the limitations of claim 15, for example, but not exclusively so, the steps of authenticating the membership verification token, establishing a connection, requesting at least one electronic identification references from the at least one communications console, and branding metadata.

76. EWS's customers, through their purchase and/or use of the Zelle® network, have not and are not infringing upon any claims of the '555 Patent because, among other reasons: (i) the claims of the '555 Patent are invalid and/or unenforceable; and (ii) EWS's customers do not meet all of the limitations of claim 15, for example, but not exclusively so, the steps of requesting at least one electronic identification references from the at least one communications console, receiving the at least one electronic identification reference from the at least one communications console; and branding metadata.

**COUNT 1**  
**(Invalidity under 35 U.S.C. § 101)**

77. EWS realleges and incorporates paragraphs 1 through 76 as if fully set forth in this paragraph.

78. As a result of the acts described in the foregoing paragraphs, a substantial controversy of sufficient immediacy and reality exists to warrant the issuance of a declaratory judgment.

79. A judicial declaration is necessary and appropriate so that EWS may ascertain its rights regarding the '555 Patent.



80. Upon information and belief, all of the claims of the '555 Patent are invalid under 35 U.S.C. § 101 for one or more of the following reasons, as well as others hereinafter set forth or which EWS may hereafter discover or otherwise become informed:

(a) The alleged invention(s) are directed to an abstract idea and contain no inventive concept sufficient to transform the claimed abstract idea into a patent-eligible application.

81. EWS is entitled to a declaratory judgment that the claims of the '555 Patent are invalid under 35 U.S.C. § 101.

**COUNT 2**  
**(Invalidity under 35 U.S.C. §§ 102 and/or 103 (pre-AIA))**

82. EWS realleges and incorporates paragraphs 1 through 81 as if fully set forth in this paragraph.

83. Grecia alleges in his demand letter that EWS and its customers have infringed and are infringing the '555 Patent. EWS denies that it or its customers in any way infringe the '555 Patent.

84. As a result of the acts described in the foregoing paragraphs, a substantial controversy of sufficient immediacy and reality exists with respect to the validity of the '555 Patent to warrant the issuance of a declaratory judgment.

85. A judicial declaration is necessary and appropriate so that EWS may ascertain its rights regarding the '555 Patent.

86. Upon information and belief, all of the claims of the '555 Patent are invalid under 35 U.S.C. §§ 102 and/or 103 (pre-AIA) for one or more of the following reasons, as well as others hereinafter set forth or which EWS may hereafter discover or otherwise become informed:

(a) The claimed invention(s) was/were patented or described in a printed publication in this for a foreign country, or were in public use, on sale or sold in this country, more than one year prior to the date of the application thereof in the United States; and/or

(b) The difference between the subject matter sought to be patented in the '555 Patent and the prior art is such that the subject matter as a whole would have been obvious at the time the alleged invention(s) was/were made to a person having ordinary skill in the art to which said subject matter pertains.

87. Each and every limitation of the claims of the '555 Patent, and specifically at least claim 23 that Grecia explicitly alleges as being infringed, is disclosed in the prior art references, including but not limited to, in the Levy reference. For example, but not exclusively so, claims 1-8, 10-13, and 15-26 are anticipated by Levy.

88. The claims of the '555 Patent are rendered obvious by the combined disclosures and teachings of the prior art, including SAML and Applied Cryptography. For example, but not exclusively so, claims 2 and 11 are rendered obvious by Levy in view of SAML and claims 9 and 14 are rendered obvious by Levy in view of Applied Cryptography.

89. EWS is entitled to a declaratory judgment that the claims of the '555 Patent are invalid under 35 U.S.C. §§ 102 and/or 103 (pre-AIA).

**COUNT 3**  
**(Invalidity under 35 U.S.C. § 112 (pre-AIA))**

90. EWS realleges and incorporates paragraphs 1 through 89 as if fully set forth in this paragraph.

91. As a result of the acts described in the foregoing paragraphs, a substantial controversy of sufficient immediacy and reality exists to warrant the issuance of a declaratory judgment.

92. A judicial declaration is necessary and appropriate so that EWS may ascertain its rights regarding the '555 Patent.

93. Upon information and belief, all of the claims of the '555 Patent are invalid under 35 U.S.C. § 112 (pre-AIA) for one or more of the following reasons, as well as others hereinafter set forth or which EWS may hereafter discover or otherwise become informed:

(a) The specification of the '555 Patent does not contain a written description of the alleged invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same; and/or

(b) The specification of the '555 Patent does not conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the inventor regards as his alleged invention.

94. EWS is entitled to a declaratory judgment that the claims of the '555 Patent are invalid under 35 U.S.C. § 112 (pre-AIA).

**COUNT 4**  
**(Unenforceability for Fraud on the Patent Office)**

95. EWS realleges and incorporates paragraphs 1 through 94 as if fully set forth in this paragraph.

96. As a result of the acts described in the foregoing paragraphs, a substantial controversy of sufficient immediacy and reality exists to warrant the issuance of a declaratory judgment.

97. A judicial declaration is necessary and appropriate so that EWS may ascertain its rights regarding the '555 Patent.

98. Upon information and belief, Grecia knowingly and wilfully committed fraud on the Patent Office by making false representations with an intent to defraud the USPTO into issuing Certificate of Corrections for the '555 Patent.

99. But for Grecia's false statements, the Patent Office would not have issued Certificates of Corrections for the '555 Patent.

100. Because Grecia obtained the Certificate of Corrections through fraud on the USPTO, the claims of the '555 Patent are unenforceable.

101. EWS is entitled to a declaratory judgment that that the claims of the '555 Patent are unenforceable due to fraud on the USPTO.

**COUNT 5**  
**(Declaration of No Patent Infringement)**

102. EWS realleges and incorporates paragraphs 1 through 101 as if fully set forth in this paragraph.

103. Any claims of the '555 Patent that may not, *arguendo*, be held invalid and/or unenforceable are so restricted in scope that EWS has not infringed, and does not infringe, upon any such claims.

104. Any claims of the '555 Patent that may not, *arguendo*, be held invalid and/or unenforceable are so restricted in scope that EWS's customers have not infringed, and do not infringe, upon any such claims.

105. As a result of the acts described in the foregoing paragraphs, a substantial controversy of sufficient immediacy and reality exists to warrant the issuance of a declaratory judgment.

106. EWS is entitled to a declaratory judgment that it has not infringed and does not infringe, directly or indirectly, any valid and enforceable claim of the '555 Patent.

107. EWS is entitled to a declaratory judgment that its customers, through their purchase and/or use of the Zelle® network, have not infringed and do not infringe, directly or indirectly, any valid and enforceable claim of the '555 Patent.

### **PRAYER FOR RELIEF**

WHEREFORE, EWS, reserving its right to amend its pleadings to add additional defense, affirmative defences, and counterclaims if warranted by discovery, respectfully requests that the Court grant the following relief:

- (a) Judgment in EWS's favor on each Count;
- (b) A declaration from the Court that the claims of the '555 Patent are invalid;
- (c) A declaration from the Court that the '555 Patent is unenforceable due to fraud on the USPTO;
- (d) A declaration from the Court that EWS and its Zelle® network do not indirectly infringe, e.g., do not induce or contributorily infringe, any claim of the '555 Patent;
- (e) A declaration from the Court that EWS and its Zelle® network do not directly infringe any claim of the '555 Patent;
- (f) A declaration from the Court that EWS's customers, by using EWS's Zelle® network, do not indirectly infringe, e.g., do not induce or contributorily infringe, any claim of the '555 Patent;
- (g) A declaration from the Court that EWS's customers, by using EWS's Zelle® network, do not directly infringe any claim of the '555 Patent;
- (h) A judgment that Grecia is not entitled to damages, injunctive or other relief in this or any future action related to any claim of the '555 Patent filed against EWS or its customers;

(i) An order declaring that EWS is a prevailing party and that this is an exceptional case, awarding EWS its costs, expenses, and reasonable attorneys' fees under 35 U.S.C. § 285; and

(j) That EWS be granted such other and additional relief as the Court deems just and proper.

Dated: March 4, 2021

RESPECTFULLY SUBMITTED,

**BAKER & HOSTETLER LLP**

By: /s/ Daniel J. Goettle  
Daniel J. Goettle  
Cira Centre, 12<sup>th</sup> Floor  
2929 Arch Street  
Philadelphia, PA 19104  
Phone:  
Facsimile:  
Email:

*Attorneys for Plaintiff Early Warning Services, LLC*

# EXHIBIT 1



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		
<p>The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.</p> <p>This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.</p>			

**Secrecy Order 37 CFR 5.2**

<input type="checkbox"/>	Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)
--------------------------	---

**Inventor Information:**

Inventor 1 <span style="float: right;">Remove</span>				
Legal Name				
Prefix	Given Name	Middle Name	Family Name	Suffix
Mr.	William		Grecia	
Residence Information (Select One) <input checked="" type="radio"/> US Residency <input type="radio"/> Non US Residency <input type="radio"/> Active US Military Service				
City	Downingtown	State/Province	PA	Country of Residence US
Mailing Address of Inventor:				
Address 1	2885 Sanford Ave SW #13208			
Address 2				
City	Grandville	State/Province	MI	
Postal Code	49418	Country	USA	
All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the <b>Add</b> button. <span style="float: right;">Add</span>				

**Correspondence Information:**

Enter either Customer Number or complete the Correspondence Information section below. For further information see 37 CFR 1.33(a).			
<input type="checkbox"/> An Address is being provided for the correspondence information of this application.			
Customer Number	70984		
Email Address	sa.cs2cd@gmail.com	Add Email	Remove Email
Email Address	cs2cd@yahoo.com	Add Email	Remove Email

**Application Information:**

Title of the Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		
Attorney Docket Number		Small Entity Status Claimed	<input type="checkbox"/>
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	7	Suggested Figure for Publication (if any)	3

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		

**Filing By Reference :**

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

**Publication Information:**

☒ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☐ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

**Representative Information:**

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	<input checked="" type="radio"/> Customer Number	<input type="radio"/> US Patent Practitioner	<input type="radio"/> Limited Recognition (37 CFR 11.9)
Customer Number	70984		

**Domestic Benefit/National Stage Information:**

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, or 365(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.

When referring to the current application, please leave the application number blank.

Prior Application Status		<a href="#">Remove</a>			
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)		
	<u>Continuation of</u>	<u>13740086</u>	<u>2013-05-06</u>		
Prior Application Status	Patented	<a href="#">Remove</a>			
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
13740086	Continuation of	13397517	2013-01-11	8533860	2013-09-10

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		

Prior Application Status	Patented		<a href="#">Remove</a>		
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
13397517	Continuation of	12985351	2012-02-15	8402555	2013-03-19

Prior Application Status	Abandoned		<a href="#">Remove</a>		
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)		
12985351	Continuation of	12728218	2011-01-06		
Prior Application Status	Abandoned		<a href="#">Remove</a>		
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)		
12728218			2010-03-21		

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.

## Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55(d). When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)<sup>i</sup> the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(h)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

<a href="#">Remove</a>			
Application Number	Country <sup>i</sup>	Filing Date (YYYY-MM-DD)	Access Code <sup>i</sup> (if applicable)

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

## Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.

☐ NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		

## Authorization to Permit Access:

☒ Authorization to Permit Access to the Instant Application by the Participating Offices

If checked, the undersigned hereby grants the USPTO authority to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the World Intellectual Property Office (WIPO), and any other intellectual property offices in which a foreign application claiming priority to the instant patent application is filed access to the instant patent application. See 37 CFR 1.14(c) and (h). This box should not be checked if the applicant does not wish the EPO, JPO, KIPO, WIPO, or other intellectual property office in which a foreign application claiming priority to the instant patent application is filed to have access to the instant patent application.

In accordance with 37 CFR 1.14(h)(3), access will be provided to a copy of the instant patent application with respect to: 1) the instant patent application-as-filed; 2) any foreign application to which the instant patent application claims priority under 35 U.S.C. 119(a)-(d) if a copy of the foreign application that satisfies the certified copy requirement of 37 CFR 1.55 has been filed in the instant patent application; and 3) any U.S. application-as-filed from which benefit is sought in the instant patent application.

In accordance with 37 CFR 1.14(c), access may be provided to information concerning the date of filing this Authorization.

## Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

### Applicant 1

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

Clear

☐ Assignee
 ☐ Legal Representative under 35 U.S.C. 117
 ☐ Joint Inventor

☐ Person to whom the inventor is obligated to assign.
 ☐ Person who shows sufficient proprietary interest

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor :

If the Applicant is an Organization check here. ☐

Prefix	Given Name	Middle Name	Family Name	Suffix

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		

<b>Mailing Address Information For Applicant:</b>			
Address 1			
Address 2			
City		State/Province	
Country		Postal Code	
Phone Number		Fax Number	
Email Address			
Additional Applicant Data may be generated within this form by selecting the Add button.			

## Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

<b>Assignee 1</b>				
Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.				
If the Assignee or Non-Applicant Assignee is an Organization check here. <input type="checkbox"/>				
Prefix	Given Name	Middle Name	Family Name	Suffix
<b>Mailing Address Information For Assignee including Non-Applicant Assignee:</b>				
Address 1				
Address 2				
City		State/Province		
Country		Postal Code		
Phone Number		Fax Number		
Email Address				
Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button.				

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

<b>Application Data Sheet 37 CFR 1.76</b>		Attorney Docket Number	
		Application Number	
Title of Invention	DIGITAL CLOUD ACCESS (PDMAS PART III)		

## Signature:

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications.					
<b>Signature</b>	/william grecia/			<b>Date (YYYY-MM-DD)</b>	2014-07-22
<b>First Name</b>	William	<b>Last Name</b>	Grecia	<b>Registration Number</b>	70984
Additional Signature may be generated within this form by selecting the Add button.					

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

## Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.



# EXHIBIT 2

**IN THE UNITED STATES DISTRICT COURT  
FOR THE WESTERN DISTRICT OF TEXAS  
WACO DIVISION**

**WILLIAM GRECIA,**

**Plaintiff,**

**vs.**

**SAMSUNG AUSTIN  
SEMICONDUCTOR, LLC**

**Defendant.**

**Civil Action File No.: 6:21-cv-00131**

**JURY TRIAL DEMANDED**

**COMPLAINT FOR WILLFUL PATENT INFRINGEMENT**

William Grecia brings this patent-infringement action against Samsung Austin Semiconductor, LLC (“Samsung”).

**Nature of the Action**

William Grecia built a product that monitors access to digital content. His product is called “Digital Debit,” and Mr. Grecia’s company has registered the “Digital Debit” trademark.

Mr. Grecia also patented his product. Since 2014, Mr. Grecia has been trying, thorough litigation and otherwise, to license his patents to Samsung.

This case is about a computer product called Samsung Knox.

Samsung Knox monitors access to digital content stored in Samsung customers’ accounts.

Samsung has willfully infringed Mr. Grecia’s patent for years and owes Mr. Grecia a reasonable royalty.

### **Parties**

1. Plaintiff William Grecia is an individual residing in Downingtown, Pennsylvania.
2. Defendant Samsung Austin Semiconductor, LLC is a Delaware limited liability company with its principal place of business at 12100 Samsung Boulevard, Austin, Texas 78754.

### **Jurisdiction and Venue**

3. This action arises under the patent laws of the United States, 35 U.S.C. §§ 101 *et seq.*
4. This Court has subject matter jurisdiction over this action under 28 U.S.C. §§ 1331 and 1338(a).
5. This Court may exercise personal jurisdiction over Samsung. Samsung conducts continuous and systematic business in this District; and this patent-infringement case arises directly from Samsung's continuous and systematic activity in this District. In short, this Court's exercise of jurisdiction over Samsung would be consistent with the Texas long-arm statute and traditional notions of fair play and substantial justice.
6. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391(b)(1)-(2) and 1400(b).

### **Infringement of U.S. Patent No. 8,402,555**

7. William Grecia is the exclusive owner of United States Patent No. 8,402,555 (the "'555 patent"), attached hereto as "Exhibit A."
8. The '555 patent is valid and enforceable.
9. Samsung infringes claim 17 of the '555 patent. Samsung makes, uses, and sells the Samsung Knox computer product. This product associates customer account's with a Knox service account identifier.

10. Attached hereto as “Exhibit B” and incorporated into this complaint as if alleged herein is a claim chart setting forth the language of claim 17 and the accused Knox computer product that Samsung makes, uses, and sells.

11. Independent claim 15 of the ’555 patent is a computer product: “A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of . . . .” (’555 patent, col. 16:46-52.)

12. Claim 17 depends from independent claim 15 and recites “the computer program product of claim 15, wherein the computer program product facilitates access rights authentication for the encrypted digital media, the branding request is an access request, and wherein the read or write request of metadata is performed in connection with a combination of a memory, CPU, server, database, and cloud system; the access request is generated by either a human user, a machine, or a human programmed computerized device; the access request further comprises a membership verification token and a rights token; wherein the rights token is a flag indicating the verification token is successfully verified.” The Samsung Knox computer program product is capable of facilitating access to Knox Enabled Apps (KEA). The Samsung Knox computer program product is capable of requesting KEA app access request generated by a machine (e.g., “Knox Platform is anchored in the actual chipset of the device... Samsung phones, tablets, and wearables”). The Samsung Knox computer program product is capable of a read or write request of KEA app metadata (required to operate, see evidence to the right) that can be performed in connection with a combination of: Knox device CPU; Knox device memory;

Knox License Manager; Knox user database, and Knox Cloud Solutions. The Samsung Knox computer program product is capable of recognizing a Samsung Knox account (membership verification token) and a Access Token (rights token). The Access Token is a flag for the Knox computer program product that the Samsung Knox is successfully verified.

13. Claim 17's product has the capacity for "authenticating the membership verification token, the authentication being performed in connection with a token database . . . ." ('555 patent, col. 16:60-62.) The Samsung Knox computer program product is capable of authenticating the Knox Account (membership verification token) with a Knox Account database (e.g., "If you enter your password incorrectly 5 consecutive times, your account will be locked for 10 minutes").

14. Samsung Knox is a computer program product for use with Samsung devices (e.g., computers) that is "tied" to operating system code stored in Samsung device storage (see page 1 "chipset" admission evidence). The Samsung Knox computer program product is capable of receiving an access read or write request of Knox encrypted data associated metadata comprising a Knox Account (membership verification token) from at least one communication console of a Knox device of a plurality of over 286 Knox devices.

15. Claim language: "establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Application Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process . . . ." ('555 patent, cols. 16:63-17:2.)

16. The Samsung Knox computer program product communications console comprises a GUI (e.g., Android Knox UI) and a Knox API protocol which is related to and capable of establishing a connection to the Knox Web Services.

17. Claim 17 is a product with the capacity of “requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user; receiving the at least one electronic identification reference from the at least one communications console . . . .” (’555 patent, col. 17:3-8.)

18. The Samsung Knox computer program product is capable of requesting and receiving a Knox Web Service account identifier (IMEI/MEID).

19. Claim 17’s product includes the capacity for “branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.” Samsung Knox computer program product is capable of writing the Knox Account (membership verification token) and the Knox Web Service web service account identifier (IMEI/MEID) into the Knox metadata (e.g., “metadata collected for every flow that is created on the device for sending and receiving application data).

20. Samsung’s infringement of the ’555 patent is willful. Attached hereto as “Exhibit C,” “Exhibit D,” and “Exhibit E,” is correspondence regarding Samsung’s need for a license under the William Grecia patents, including the ’555 patent.

### **Prayer for Relief**

WHEREFORE, William Grecia prays for the following relief against Samsung:

- (a) Judgment that Samsung has directly infringed claim 17 of the ’555 patent;
- (b) For a reasonable royalty;

- (c) For pre-judgment interest and post-judgment interest at the maximum rate allowed by law;
- (d) For injunctive relief, including a preliminary injunction; and
- (e) For such other and further relief as the Court may deem just and proper.

### **Demand for Jury Trial**

William Grecia demands a trial by jury on all matters and issues triable by jury.

Respectfully Submitted,

Date: February 6, 2021

/s/Artoush Ohanian  
Texas State Bar No. 24013260  
H. Artoush Ohanian  
[artoush@ohanianip.com](mailto:artoush@ohanianip.com)  
OHANIANIP  
604 West 13th Street  
Austin, Texas 78701  
(512) 298.2005 (telephone & facsimile)

Matthew M. Wawrzyn (application for *pro hac vice* admission forthcoming)  
[matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)  
WAWRZYN LLC  
200 East Randolph Street, Suite 5100  
Chicago, IL 60601  
(312) 235-3120 (telephone)  
(312) 233-0063 (facsimile)

*Counsel for William Grecia*

# EXHIBIT 3



---

**From:** Matt Wawrzyn <matt@wawrzynlaw.com>  
**Sent:** Wednesday, December 23, 2020 3:59 PM  
**To:** Chen, George  
**Subject:** Fwd: All Four (4) 2021 cases - Zelle - Western District Jurisdiction  
**Attachments:** FNBCT\_Zelle\_555-23.pdf; Frost Bank\_Zelle\_555-23.pdf; American-Bank\_Zelle\_555-23.pdf; FNBT\_Zelle\_555-23.pdf

George:

Matt

----- Forwarded message -----

**From:** **Grecia Family**  
**Date:** Wed, Dec 23, 2020 at 4:04 AM  
**Subject:** All Four (4) 2021 cases - Zelle - Western District Jurisdiction  
**To:** Matt Wawrzyn <[matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)>

Matt:

Forgive the multiple emails while forming this information together. As expressed in the earlier email, Grecia will be enforcing only 4 new cases in 2021, all in Albright's jurisdiction. These are the same 4 cases Aliaswire settled in Albright court, but with a much less and weaker set of patent claims than 555.

These are the 4 cases EWS are welcome to buy licenses for as the indemnifying party for a early discounted amount of \$600,000 each case. After filing, each case will seek to settle for \$1.2mln each.

Albright respects the OOVOO CAFC precedent and these cases will hit Discovery.

In Albright's discovery, Grecia will demand and get access to each of the 4 cases mobile source codes. It's advisable that EWS consider their choices on their past IPR success and move wisely.

EWS has until January 5 to either buy out these cases early or Grecia moves forward. Buyout for all 4 cases would total \$2.4mln but will exit higher than the JPMC disputed agreement amount per new Zelle case.

Grecia will not consider any bulk or global Zelle deals at all in 2021. If JPMC cures their breach in 2021 then Grecia may consider revisiting a global Zelle deal in the future after time has passed to allow the JPMC damage to any good faith between the parties to heal.

For now, these are the 4 2021 cases that will be enforced and filed with Albright directly after January 5th 2021.

These 4 charts with the above avoidance/license offer are requested to go out today to EWS, If unable to accept these for any possible reason should be informed back to us by Dec 28.

\*Only Grecia's IP portfolio with Re-Issue license is offered, Qondado and WGP LLC assets are not inclusive or offered.

Thank You,

*Grecia Estate & Trust*

Grecia Family Estate & Trust

**P:** (212) 390-0355

**E:** business@greciafamily.estate

CONFIDENTIALITY NOTICE: The contents of this email message and any attachments are intended solely for the addressee(s) and may contain confidential and/or privileged information and may be legally protected from disclosure. If you are not the intended recipient of this message or their agent, or if this message has been addressed to you in error, please immediately alert the sender by reply email and then delete this message and any attachments. If you are not the intended recipient, you are hereby notified that any use, dissemination, copying, or storage of this message or its attachments is strictly prohibited.

--

Matthew M. Wawrzyn

[wawrzynllc.com](http://wawrzynllc.com)

200 East Randolph Street

Suite 5100

Chicago, IL 60601

312.235.3120 (office)

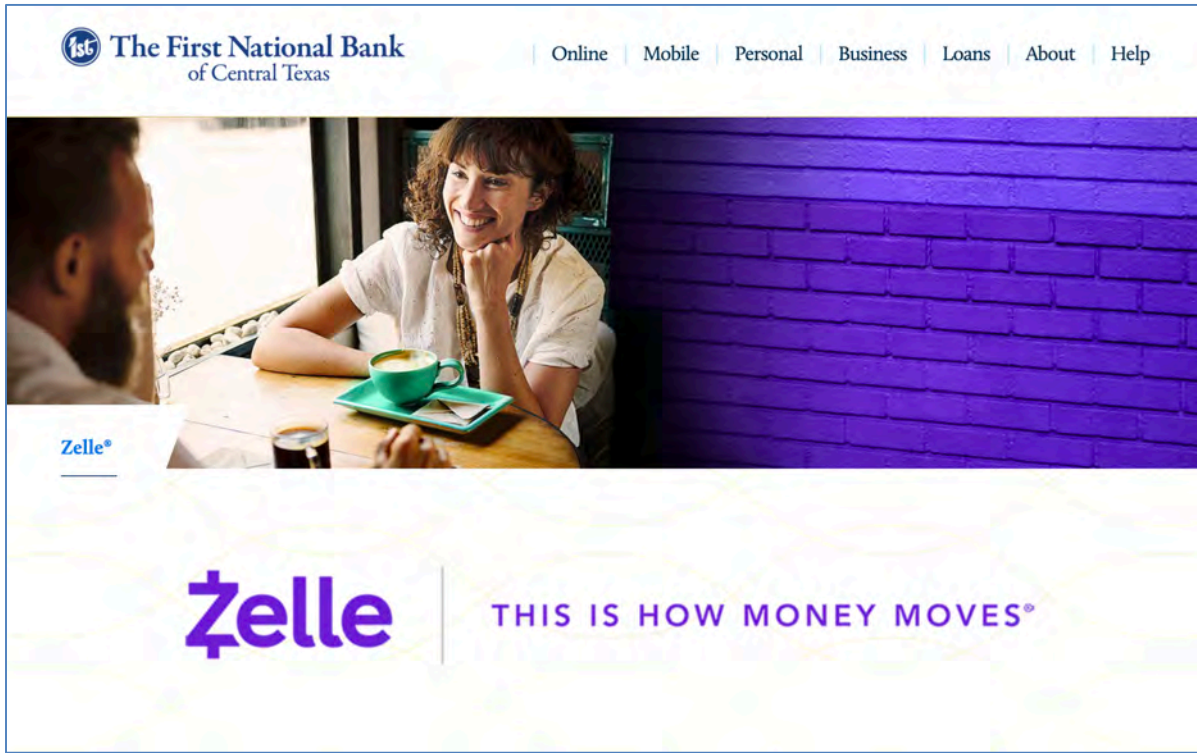
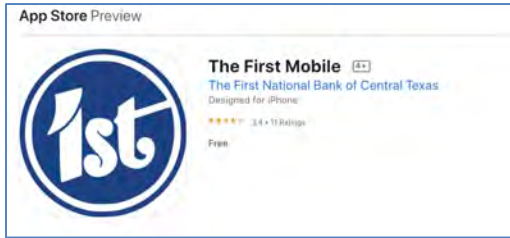
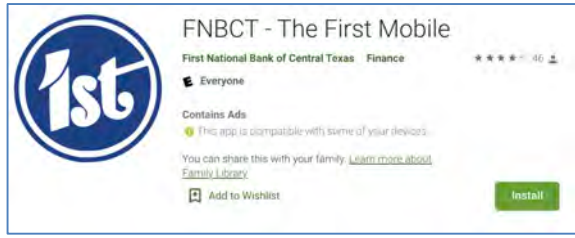
312.233.0063 (facsimile)

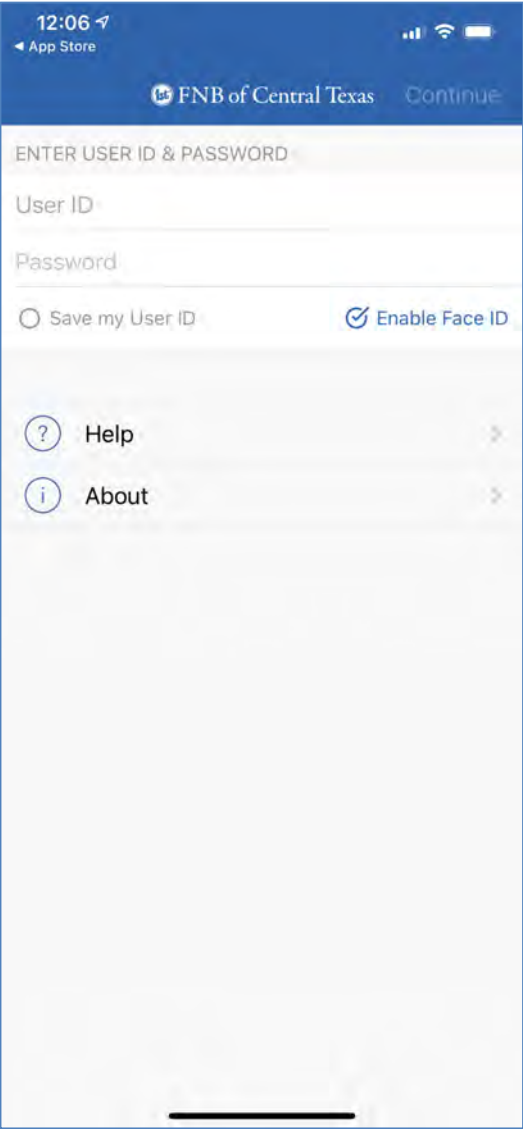
847.274.9844 (mobile)

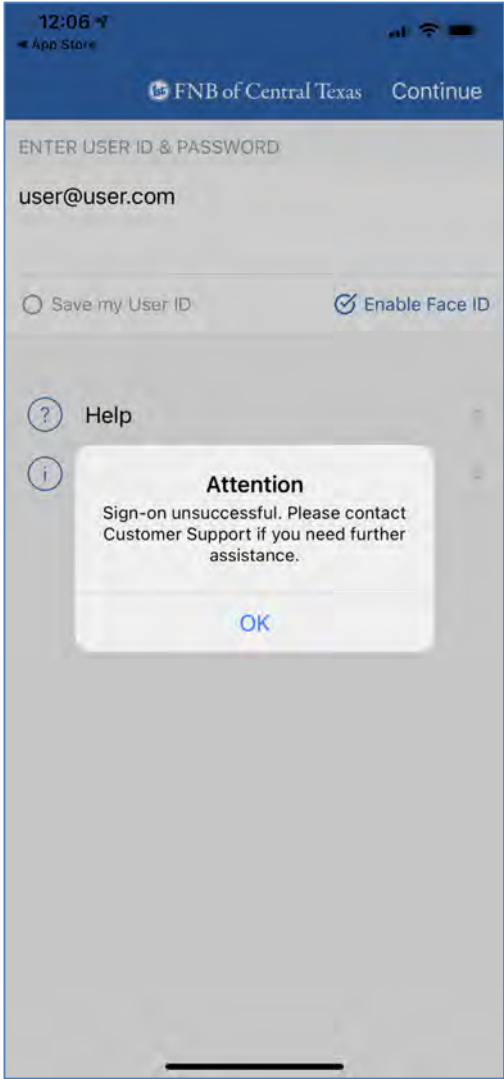
[matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)

**U.S. Patent #8,402,555: FNBCT With Zelle**

Computer Readable Medium Claim (Mobile App)

Claim 23	Analysis	Select Evidence
<p>The computer program product according to claim 15, wherein the encrypted digital media [Sullivan claim construction order meaning: computer facilitated data] is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.</p> <p>A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:</p>	<p>The FNBCT mobile application is a computer program product for use with a (mobile) computer to associate FNBCT accounts with a Zelle service account identifier for cross-reference by writing and cross-referencing a Zelle web service CXCToken (see page 6) to the FNBCT computer product metadata.</p>	 <p>Safely send and receive money with <i>Zelle</i> through The First Mobile banking app.</p> <div>   </div> <p>Source: Apple App Store</p> <p>Source: Google Play Store</p> <p>Source: <a href="https://www.fnbct.com/getzelle/">https://www.fnbct.com/getzelle/</a></p>

<p>receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;</p>	<p>FNBCT computer product receives an access request by receiving a membership verification token through the FNBCT computer product's communication console.</p>	
---	---	---

<p>authenticating the membership verification token, the authentication being performed in connection with a token database;</p>	<p>FNBCT computer product authenticate the membership verification token with a User ID database.</p>	
--	---	---

establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

FNBCT computer product establish an API communication related to the ZELLE RESTful API.

"We expose our API system to the banks when they join our network. At this point we are focused on delivering the service to the banks so that we can have the most robust and secure network available. In the future we may look at opening our APIs more broadly, but at this point we are exposing them to the banks who are the part of the network", said Mike.

#### About Early Warning

Early Warning provides risk management solutions to a diverse network of 2,300 financial institutions, government entities and payment companies, enabling businesses and consumers to transact securely and conveniently. Owned and governed by Bank of America, BB&T, Capital One, JPMorgan Chase and Wells Fargo, Early Warning's unique business model facilitates a data exchange system based on collaborative, shared intelligence. For 25 years, the company has worked with organizations of all sizes to advance collaborative risk management and fraud prevention. For more information please [visit](#).

Source: <https://letstalkpayments.com/interview-with-early-warning-and-clearxchange-the-most-powerful-bank-focused-alliance-in-the-authentication-and-digital-payments-industry/>

## ZELLE RESTful API DOCUMENTATION

organization IDs.	
<i>participantID</i>	<p>A 64-byte string used to uniquely identify a Zelle participant.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the Zelle Shared Directory. This value may or may not be the same as the partner ID of the Zelle participant. When possible, it is recommended that the participant ID match the partner ID.</p>
<i>partnerID</i>	<p>A 20-byte string used to uniquely identify a Zelle participant in FTM.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the FTM Shared Directory. This value may or may not be the same as the participant ID of the Zelle participant. When possible, it is recommended that the partner ID match the participant ID.</p>

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>  
Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>



requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console;

FNBCT computer product request and receive a CXCToken (e.g., an account electronic Identification reference) from the ZELLE RESTful API communication.

#### WEB SERVICES

##### CREATE (POST)

This service returns the location (URL) of the newly-created CXCToken with an HTTP status of 201 (CREATED) if executed successfully, or one of the HTTP 4xx error codes if an error occurs.

`https://<servername:serverport>/fxh/svc/cxctokens/`

##### DESCRIPTION

The CXCToken.Create web service registers a token for a participant present in the FTM database.

To register a new participant token, the FTM web service checks to ensure that the maximum number of active tokens has not been reached and, if not, invokes four web services. First, the web service invokes the Zelle Match Recipient to determine if the token already exists. If the token does not already exist, the CXCToken.Create web service calls the Client Service to refresh the fraud detection data provided from the user interface. Next, a call is made to the fraud application to perform fraud detection, storing the fraud check result in the FTM database. If no fraud is detected, the web service creates a payment profile ID and invokes the Zelle Change Token Status web service. For a new participant token, Zelle creates the payment profile and returns a response to FTM to indicate that the participant token was successfully added. Then, the CXCToken.Create web service updates the payment profile and associated token to the FTM database and returns a response to the user interface to indicate that the participant token was successfully added. FTM then calls the Client Service to notify the participant of successful token registration.

24

© Copyright IBM Corp. 2017,2018.

^Page 24:

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

#### CXCTOKEN

A CXCToken entity represents a Zelle token and its associated Zelle payment profile. The base URL for this service is

`https://<servername:serverport>/fxh/svc/cxctokens/`

#### ATTRIBUTES

##### KEY

partnerID : partnerType : participantID : token : paymentProfileID

^Page 21

NOTE: When creating a new token, the payment profile is generated by FTM.

##### token

A 255-byte string that is a normalized value containing either the email address or mobile phone number of a Zelle participant.

^Page 7, definition of "Token"

and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

FNBCT computer product write (e.g., enroll user to Zelle) to associate the verification token (email or mobile number) and the Zelle CXCToken for cross-reference with the computer product user account data that associates with users Zelle account data.

## **I sent money to someone & they never received it. What now?**

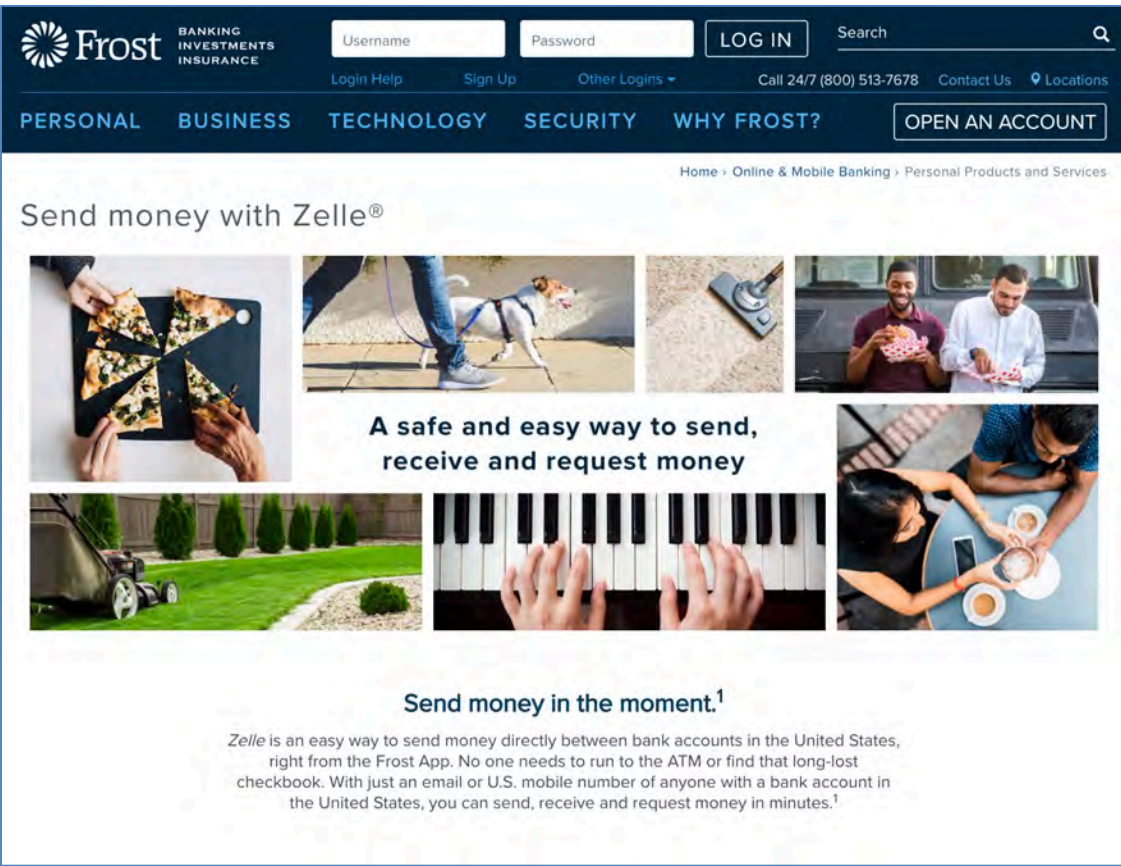
First, check the payment status within your payment activity in your bank's online or mobile service, or within the *Zelle* app. If the payment status is pending, the recipient may not have enrolled their mobile number or email address to receive the payment. If the payment status is completed, then the money is already available in the recipient's bank account, or will be within 3 business days if the recipient recently enrolled. If you aren't sure of the status of your payment, contact customer support at 254 772-9330.

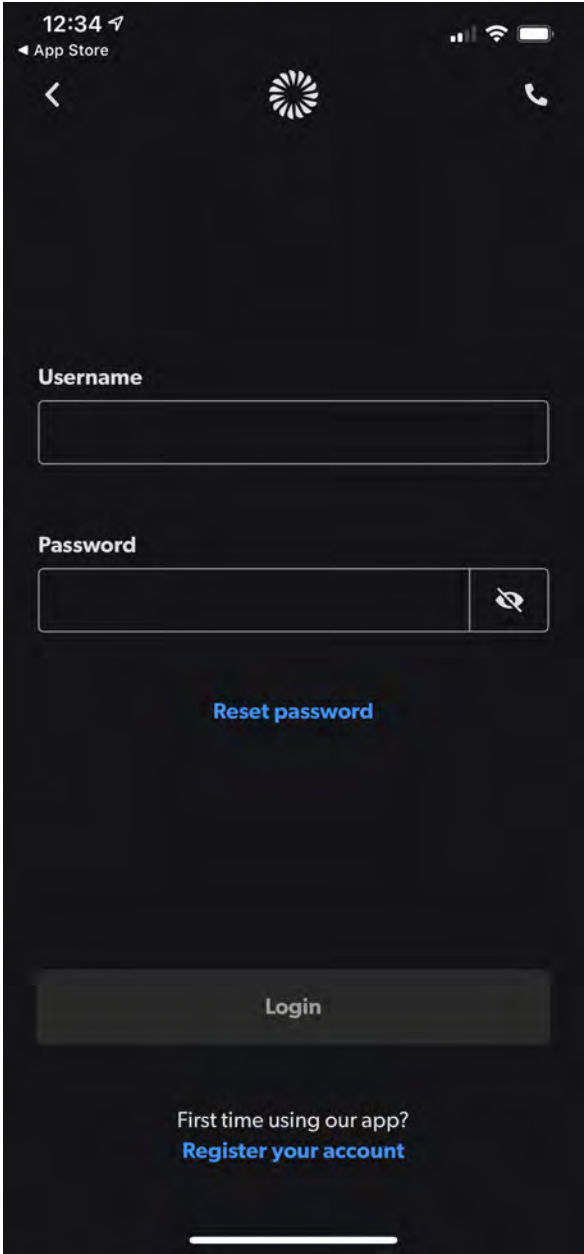
Source: <https://www.fnbct.com/getzelle/>

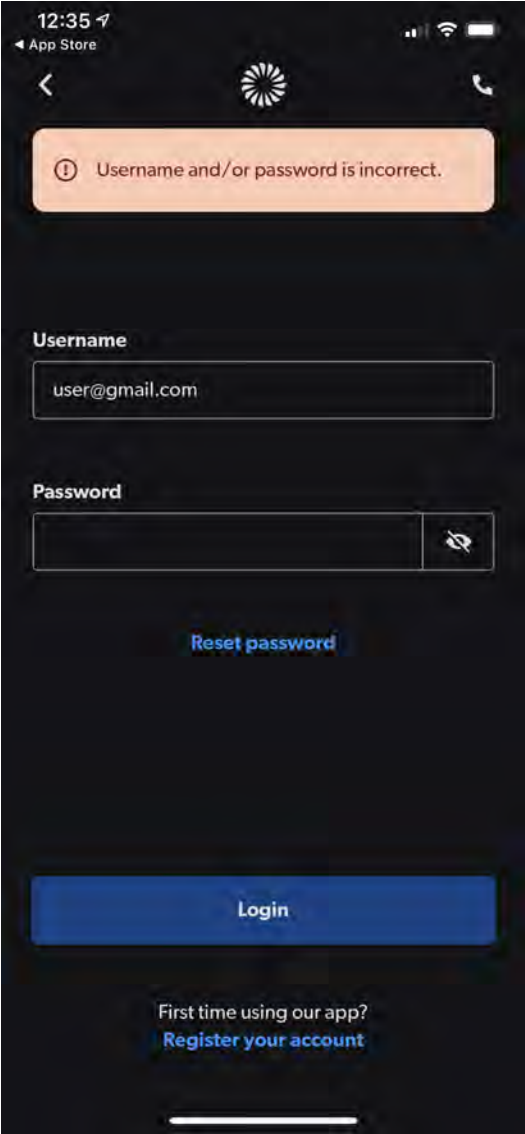


**U.S. Patent #8,402,555: FROST BANK With Zelle**

Computer Readable Medium Claim (Mobile App)

Claim 23	Analysis	Select Evidence
<p>The computer program product according to claim 15, wherein the encrypted digital media [Sullivan claim construction order meaning: computer facilitated data] is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.</p> <p>A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:</p>	<p>The FROST BANK mobile application is a computer program product for use with a (mobile) computer to associate FROST BANK accounts with a Zelle service account identifier for cross-reference by writing and cross-referencing a Zelle web service CXCToken (see page 6) to the FROST BANK computer product metadata.</p>	 <p>Source: <a href="https://www.frostbank.com/online-mobile/zelle">https://www.frostbank.com/online-mobile/zelle</a></p>

<p>receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;</p>	<p>FROST BANK computer product receives an access request by receiving a membership verification token through the FROST BANK computer product's communication console.</p>	
---	---	---

<p>authenticating the membership verification token, the authentication being performed in connection with a token database;</p>	<p>FROST BANK computer product authenticate the membership verification token with a User ID database.</p>	 <p>The screenshot shows a mobile application interface for Frost Bank. At the top, the status bar displays the time 12:35, signal strength, Wi-Fi, and battery icons. Below the status bar is a navigation bar with a back arrow, the Frost Bank logo, and a phone icon. A red error message box with a white exclamation mark icon contains the text "Username and/or password is incorrect." Below this, there are two input fields: "Username" with the text "user@gmail.com" and "Password" which is empty. To the right of the password field is a toggle icon. Below the password field is a blue link that says "Reset password". At the bottom of the screen is a large blue button labeled "Login". Below the "Login" button is the text "First time using our app?" followed by a blue link that says "Register your account".</p>
--	--	--

establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

FROST BANK computer product establish an API communication related to the ZELLE RESTful API.

"We expose our API system to the banks when they join our network. At this point we are focused on delivering the service to the banks so that we can have the most robust and secure network available. In the future we may look at opening our APIs more broadly, but at this point we are exposing them to the banks who are the part of the network", said Mike.

#### About Early Warning

Early Warning provides risk management solutions to a diverse network of 2,300 financial institutions, government entities and payment companies, enabling businesses and consumers to transact securely and conveniently. Owned and governed by Bank of America, BB&T, Capital One, JPMorgan Chase and Wells Fargo, Early Warning's unique business model facilitates a data exchange system based on collaborative, shared intelligence. For 25 years, the company has worked with organizations of all sizes to advance collaborative risk management and fraud prevention. For more information please [visit](#).

Source: <https://letstalkpayments.com/interview-with-early-warning-and-clearxchange-the-most-powerful-bank-focused-alliance-in-the-authentication-and-digital-payments-industry/>

## ZELLE RESTful API DOCUMENTATION

organization IDs.	
<i>participantID</i>	<p>A 64-byte string used to uniquely identify a Zelle participant.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the Zelle Shared Directory. This value may or may not be the same as the partner ID of the Zelle participant. When possible, it is recommended that the participant ID match the partner ID.</p>
<i>partnerID</i>	<p>A 20-byte string used to uniquely identify a Zelle participant in FTM.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the FTM Shared Directory. This value may or may not be the same as the participant ID of the Zelle participant. When possible, it is recommended that the partner ID match the participant ID.</p>

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>



requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console;

FROST BANK computer product request and receive a CXCToken (e.g., an account electronic Identification reference) from the ZELLE RESTful API communication.

#### WEB SERVICES

##### CREATE (POST)

This service returns the location (URL) of the newly-created CXCToken with an HTTP status of 201 (CREATED) if executed successfully, or one of the HTTP 4xx error codes if an error occurs.

`https://<servername:serverport>/fxh/svc/cxctokens/`

##### DESCRIPTION

The CXCToken.Create web service registers a token for a participant present in the FTM database.

To register a new participant token, the FTM web service checks to ensure that the maximum number of active tokens has not been reached and, if not, invokes four web services. First, the web service invokes the Zelle Match Recipient to determine if the token already exists. If the token does not already exist, the CXCToken.Create web service calls the Client Service to refresh the fraud detection data provided from the user interface. Next, a call is made to the fraud application to perform fraud detection, storing the fraud check result in the FTM database. If no fraud is detected, the web service creates a payment profile ID and invokes the Zelle Change Token Status web service. For a new participant token, Zelle creates the payment profile and returns a response to FTM to indicate that the participant token was successfully added. Then, the CXCToken.Create web service updates the payment profile and associated token to the FTM database and returns a response to the user interface to indicate that the participant token was successfully added. FTM then calls the Client Service to notify the participant of successful token registration.

24

© Copyright IBM Corp. 2017,2018.

^Page 24:

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

#### CXCTOKEN

A CXCToken entity represents a Zelle token and its associated Zelle payment profile. The base URL for this service is

`https://<servername:serverport>/fxh/svc/cxctokens/`

#### ATTRIBUTES

##### KEY

partnerID : partnerType : participantID : token : paymentProfileID

^Page 21

NOTE: When creating a new token, the payment profile is generated by FTM.

token A 255-byte string that is a normalized value containing either the email address or mobile phone number of a Zelle participant.

^Page 7, definition of "Token"

and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

FROST BANK computer product write (e.g., enroll user to Zelle) to associate the verification token (email or mobile number) and the Zelle CXCToken for cross-reference with the computer product user account data that associates with users Zelle account data.

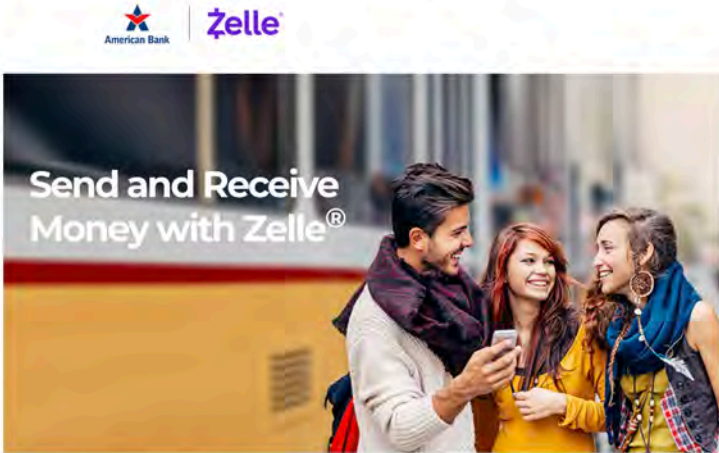


#### How long does it take to receive money with Zelle?

Money sent with Zelle is typically available to an enrolled recipient within minutes<sup>2</sup>. If you send money to someone who isn't enrolled with Zelle, they will receive a notification prompting them to enroll. After enrollment, the money will move directly to your recipient's account, typically within minutes<sup>1</sup>.

Source: <https://www.frostbank.com/online-mobile/zelle>

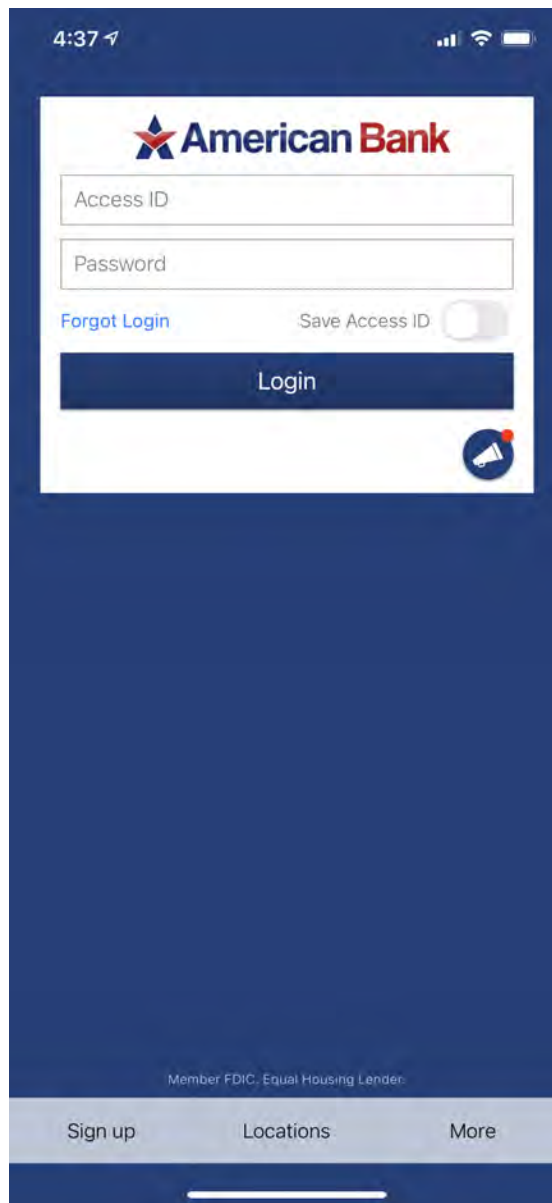
**U.S. Patent #8,402,555: AMERICAN BANK With Zelle**

Computer Readable Medium Claim (Mobile App)

Claim 23	Analysis	Select Evidence
<p>The computer program product according to claim 15, wherein the encrypted digital media [Sullivan claim construction order meaning: computer facilitated data] is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.</p> <p>A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:</p>	<p>The AMERICAN BANK mobile application is a computer program product for use with a (mobile) computer to associate AMERICAN BANK accounts with a Zelle service account identifier for cross-reference by writing and cross-referencing a Zelle web service CXCToken (see page 6) to the AMERICAN BANK computer product metadata.</p>	<div data-bbox="819 230 1606 880">  <p><b>We have partnered with Zelle to bring you a fast, safe and easy way to send and receive money with friends, family and other people you trust<sup>2</sup>.</b></p> </div> <div data-bbox="819 922 2003 1055"> <p>Zelle is available right from Online and Mobile Banking so you don't need to download anything new to start sending and receiving money!</p> </div> <div data-bbox="819 1096 1304 1359">  </div> <p>Source: Apple App Store</p> <div data-bbox="1339 1166 1862 1359">  </div> <p>Source: Google Play Store</p> <p>Source: <a href="https://www.ambankwaco.com/Zelle.aspx">https://www.ambankwaco.com/Zelle.aspx</a></p>

receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

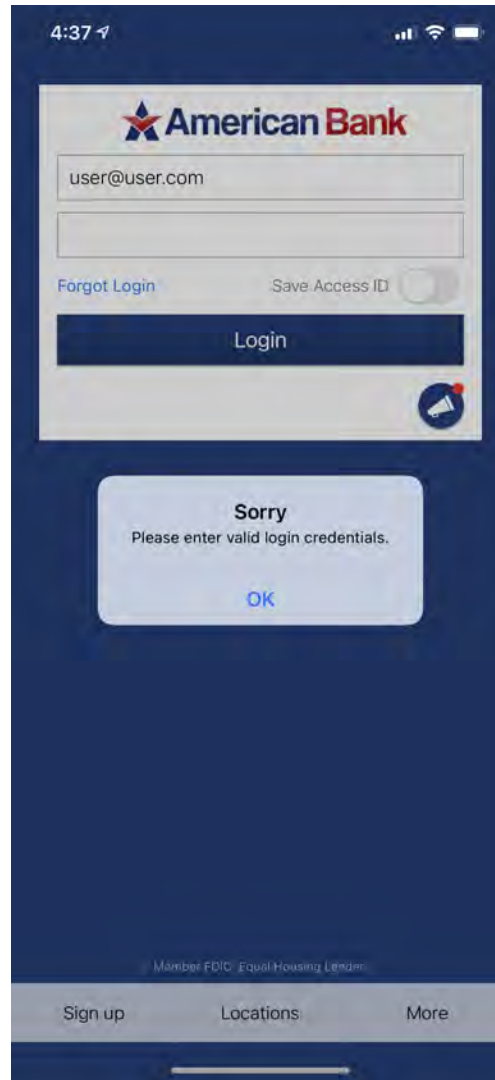
AMERICAN BANK computer product receives an access request by receiving a membership verification token through the AMERICAN BANK computer product's communication console.





authenticating the membership verification token, the authentication being performed in connection with a token database;

AMERICAN BANK computer product authenticate the membership verification token with a User ID database.



establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

AMERICAN BANK computer product establish an API communication related to the ZELLE RESTful API.

"We expose our API system to the banks when they join our network. At this point we are focused on delivering the service to the banks so that we can have the most robust and secure network available. In the future we may look at opening our APIs more broadly, but at this point we are exposing them to the banks who are the part of the network", said Mike.

#### About Early Warning

Early Warning provides risk management solutions to a diverse network of 2,300 financial institutions, government entities and payment companies, enabling businesses and consumers to transact securely and conveniently. Owned and governed by Bank of America, BB&T, Capital One, JPMorgan Chase and Wells Fargo, Early Warning's unique business model facilitates a data exchange system based on collaborative, shared intelligence. For 25 years, the company has worked with organizations of all sizes to advance collaborative risk management and fraud prevention. For more information please [visit](#).

Source: <https://letstalkpayments.com/interview-with-early-warning-and-clearxchange-the-most-powerful-bank-focused-alliance-in-the-authentication-and-digital-payments-industry/>

## ZELLE RESTful API DOCUMENTATION

organization IDs.	
<i>participantID</i>	<p>A 64-byte string used to uniquely identify a Zelle participant.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the Zelle Shared Directory. This value may or may not be the same as the partner ID of the Zelle participant. When possible, it is recommended that the participant ID match the partner ID.</p>
<i>partnerID</i>	<p>A 20-byte string used to uniquely identify a Zelle participant in FTM.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the FTM Shared Directory. This value may or may not be the same as the participant ID of the Zelle participant. When possible, it is recommended that the partner ID match the participant ID.</p>

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console;

AMERICAN BANK computer product request and receive a CXCToken (e.g., an account electronic Identification reference) from the ZELLE RESTful API communication.

#### WEB SERVICES

##### CREATE (POST)

This service returns the location (URL) of the newly-created CXCToken with an HTTP status of 201 (CREATED) if executed successfully, or one of the HTTP 4xx error codes if an error occurs.

`https://<servername:serverport>/fxh/svc/cxctokens/`

##### DESCRIPTION

The CXCToken.Create web service registers a token for a participant present in the FTM database.

To register a new participant token, the FTM web service checks to ensure that the maximum number of active tokens has not been reached and, if not, invokes four web services. First, the web service invokes the Zelle Match Recipient to determine if the token already exists. If the token does not already exist, the CXCToken.Create web service calls the Client Service to refresh the fraud detection data provided from the user interface. Next, a call is made to the fraud application to perform fraud detection, storing the fraud check result in the FTM database. If no fraud is detected, the web service creates a payment profile ID and invokes the Zelle Change Token Status web service. For a new participant token, Zelle creates the payment profile and returns a response to FTM to indicate that the participant token was successfully added. Then, the CXCToken.Create web service updates the payment profile and associated token to the FTM database and returns a response to the user interface to indicate that the participant token was successfully added. FTM then calls the Client Service to notify the participant of successful token registration.

24

© Copyright IBM Corp. 2017,2018.

^Page 24:

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

#### CXCTOKEN

A CXCToken entity represents a Zelle token and its associated Zelle payment profile. The base URL for this service is

`https://<servername:serverport>/fxh/svc/cxctokens/`

#### ATTRIBUTES

##### KEY

partnerID : partnerType : participantID : token : paymentProfileID

^Page 21

NOTE: When creating a new token, the payment profile is generated by FTM.

##### token

A 255-byte string that is a normalized value containing either the email address or mobile phone number of a Zelle participant.

^Page 7, definition of "Token"

and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

AMERICAN BANK computer product write (e.g., enroll user to Zelle) to associate the verification token (email or mobile number) and the Zelle CXCToken for cross-reference with the computer product user account data that associates with users Zelle account data.

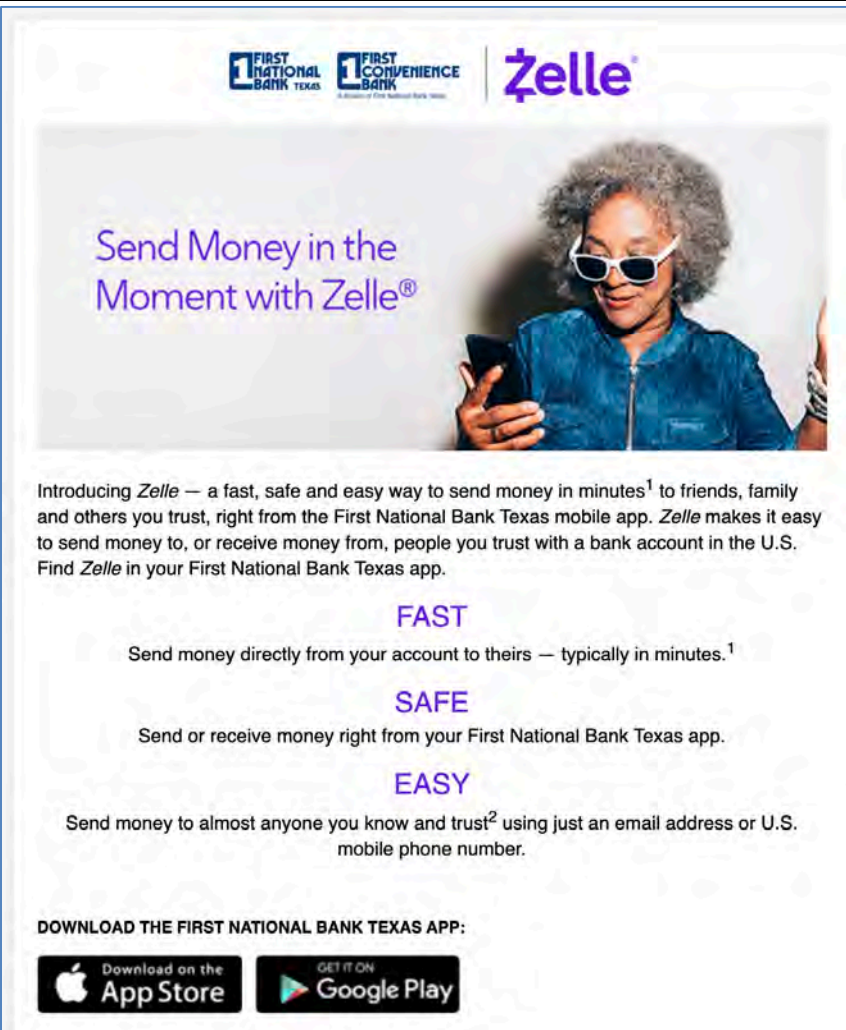
## Enroll in Zelle® Now

1. Login to **Bill Pay**
2. Select **Send Money with Zelle®**
3. Enroll your U.S. mobile number or email address
4. You're ready to start sending money with *Zelle*

Source: <https://www.ambankwaco.com/Zelle.aspx>

**U.S. Patent #8,402,555: FNBT With Zelle**

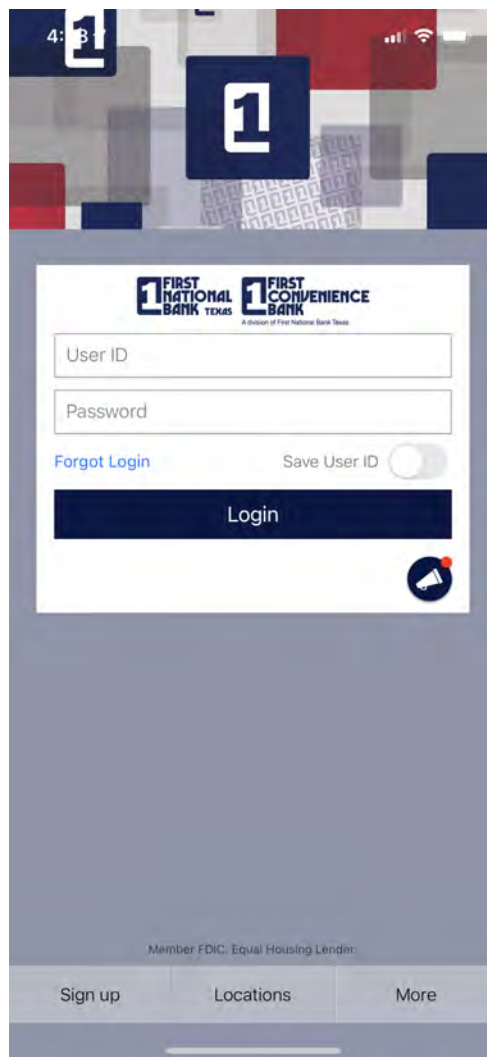
Computer Readable Medium Claim (Mobile App)

Claim 23	Analysis	Select Evidence
<p>The computer program product according to claim 15, wherein the encrypted digital media [Sullivan claim construction order meaning: computer facilitated data] is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.</p> <p>A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:</p>	<p>The FNBT mobile application is a computer program product for use with a (mobile) computer to associate FNBT accounts with a Zelle service account identifier for cross-reference by writing and cross-referencing a Zelle web service CXCToken (see page 6) to the FNBT computer product metadata.</p>	 <p>Source: <a href="https://www.1stnb.com/zelle">https://www.1stnb.com/zelle</a></p>



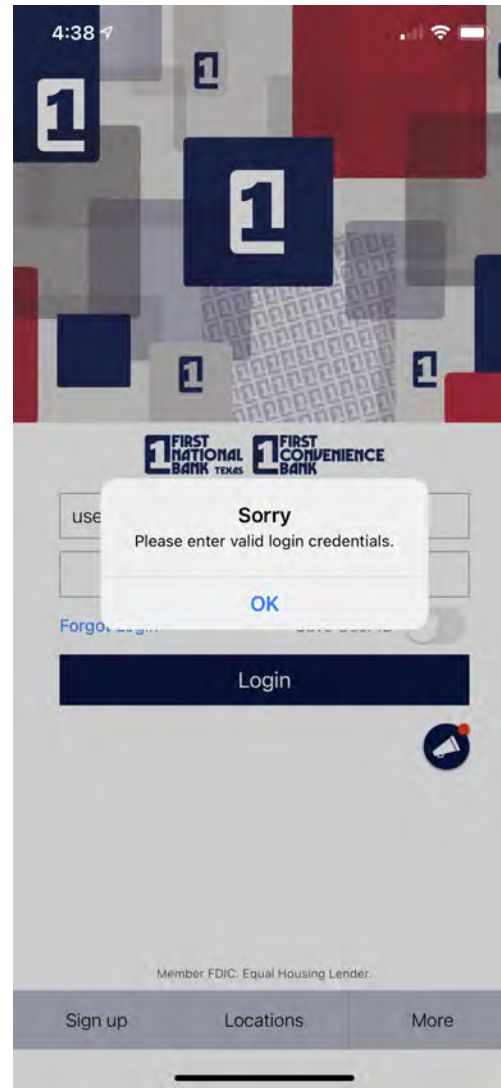
receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

FNBT computer product receives an access request by receiving a membership verification token through the FNBT computer product's communication console.



authenticating the membership verification token, the authentication being performed in connection with a token database;

FNBT computer product authenticate the membership verification token with a User ID database.



establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

FNBT computer product establish an API communication related to the ZELLE RESTful API.

"We expose our API system to the banks when they join our network. At this point we are focused on delivering the service to the banks so that we can have the most robust and secure network available. In the future we may look at opening our APIs more broadly, but at this point we are exposing them to the banks who are the part of the network", said Mike.

#### About Early Warning

Early Warning provides risk management solutions to a diverse network of 2,300 financial institutions, government entities and payment companies, enabling businesses and consumers to transact securely and conveniently. Owned and governed by Bank of America, BB&T, Capital One, JPMorgan Chase and Wells Fargo, Early Warning's unique business model facilitates a data exchange system based on collaborative, shared intelligence. For 25 years, the company has worked with organizations of all sizes to advance collaborative risk management and fraud prevention. For more information please [visit](#).

Source: <https://letstalkpayments.com/interview-with-early-warning-and-clearxchange-the-most-powerful-bank-focused-alliance-in-the-authentication-and-digital-payments-industry/>

## ZELLE RESTful API DOCUMENTATION

organization IDs.	
<i>participantID</i>	<p>A 64-byte string used to uniquely identify a Zelle participant.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the Zelle Shared Directory. This value may or may not be the same as the partner ID of the Zelle participant. When possible, it is recommended that the participant ID match the partner ID.</p>
<i>partnerID</i>	<p>A 20-byte string used to uniquely identify a Zelle participant in FTM.</p> <p><b>NOTE:</b> This value corresponds to the ID of the Zelle participant in the FTM Shared Directory. This value may or may not be the same as the participant ID of the Zelle participant. When possible, it is recommended that the partner ID match the participant ID.</p>

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>  
Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>



requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console;

FNBT computer product request and receive a CXCToken (e.g., an account electronic Identification reference) from the ZELLE RESTful API communication.

#### WEB SERVICES

##### CREATE (POST)

This service returns the location (URL) of the newly-created CXCToken with an HTTP status of 201 (CREATED) if executed successfully, or one of the HTTP 4xx error codes if an error occurs.

`https://<servername:serverport>/fxh/svc/cxctokens/`

##### DESCRIPTION

The CXCToken.Create web service registers a token for a participant present in the FTM database.

To register a new participant token, the FTM web service checks to ensure that the maximum number of active tokens has not been reached and, if not, invokes four web services. First, the web service invokes the Zelle Match Recipient to determine if the token already exists. If the token does not already exist, the CXCToken.Create web service calls the Client Service to refresh the fraud detection data provided from the user interface. Next, a call is made to the fraud application to perform fraud detection, storing the fraud check result in the FTM database. If no fraud is detected, the web service creates a payment profile ID and invokes the Zelle Change Token Status web service. For a new participant token, Zelle creates the payment profile and returns a response to FTM to indicate that the participant token was successfully added. Then, the CXCToken.Create web service updates the payment profile and associated token to the FTM database and returns a response to the user interface to indicate that the participant token was successfully added. FTM then calls the Client Service to notify the participant of successful token registration.

24

© Copyright IBM Corp. 2017,2018.

^Page 24:

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

#### CXCTOKEN

A CXCToken entity represents a Zelle token and its associated Zelle payment profile. The base URL for this service is

`https://<servername:serverport>/fxh/svc/cxctokens/`

#### ATTRIBUTES

##### KEY

partnerID : partnerType : participantID : token : paymentProfileID

^Page 21

NOTE: When creating a new token, the payment profile is generated by FTM.

##### token

A 255-byte string that is a normalized value containing either the email address or mobile phone number of a Zelle participant.

^Page 7, definition of "Token"

and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

FNBT computer product write (e.g., enroll user to Zelle) to associate the verification token (email or mobile number) and the Zelle CXCToken for cross-reference with the computer product user account data that associates with users Zelle account data.

### DOWNLOAD THE FIRST NATIONAL BANK TEXAS APP:



### GETTING STARTED IS EASY.

- 1 Log into the First National Bank Texas app.
- 2 Select Send money with Zelle®.
- 3 Enroll your U.S. mobile number or email address.
- 4 Send money to friends family and others you know and trust.

Source: <https://www.1stnb.com/zelle>

# EXHIBIT 4



US008402555B2

(12) **United States Patent**  
**Grecia**

(10) **Patent No.:** **US 8,402,555 B2**  
(45) **Date of Patent:** **Mar. 19, 2013**

(54) **PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)**

(76) Inventor: **William Grecia**, Grandville, MI (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/397,517**

(22) Filed: **Feb. 15, 2012**

(65) **Prior Publication Data**

US 2012/0151220 A1 Jun. 14, 2012

5,883,955 A	3/1999	Ronning	
5,887,060 A	3/1999	Ronning	
5,903,647 A	5/1999	Ronning	
5,907,617 A	5/1999	Ronning	
6,385,596 B1	5/2002	Wiser	
6,611,812 B2	8/2003	Hurtado	
6,665,797 B1	12/2003	Keung	
6,799,165 B1	9/2004	Boesjes	
7,254,235 B2 *	8/2007	Boudreault et al.	380/239
7,266,839 B2	9/2007	Bowers et al.	
7,290,699 B2	11/2007	Reddy	
7,340,769 B2	3/2008	Baughner	
7,343,014 B2 *	3/2008	Sovio et al.	380/278
7,386,513 B2	6/2008	Lao	
7,515,710 B2	4/2009	Grab	
7,516,495 B2	4/2009	Shoemaker	
7,526,650 B1 *	4/2009	Wimmer	713/176

(Continued)

#### Related U.S. Application Data

(63) Continuation of application No. 12/985,351, filed on Jan. 6, 2011, which is a continuation of application No. 12/728,218, filed on Mar. 21, 2010, now abandoned.

#### FOREIGN PATENT DOCUMENTS

EP	1505530 A1	2/2005
EP	1564621 A1	8/2005

(Continued)

#### OTHER PUBLICATIONS

Liu et al. 2004 NPL—A license-sharing scheme in Digital Rights Management.\*

(Continued)

(51) **Int. Cl.**  
**H04L 29/06** (2006.01)

(52) **U.S. Cl.** ..... **726/29**; 726/28; 713/185

(58) **Field of Classification Search** ..... 726/1–21,  
726/26–33; 713/155–159, 168, 172–176,  
713/185, 182

See application file for complete search history.

*Primary Examiner* — Jung Kim  
*Assistant Examiner* — Tri Tran

(56) **References Cited**

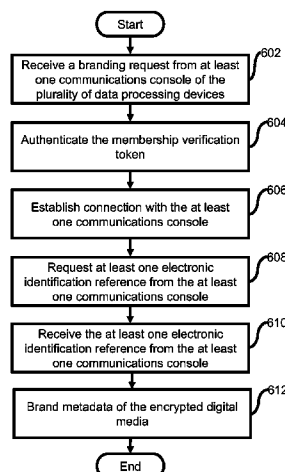
#### U.S. PATENT DOCUMENTS

5,010,571 A	4/1991	Katznelson
5,247,575 A	9/1993	Sprague
5,267,313 A	11/1993	Hirata
5,319,705 A	6/1994	Halter
5,349,642 A	9/1994	Kingdon
5,509,074 A	4/1996	Choudhury
5,586,186 A	12/1996	Yuval
5,719,938 A	2/1998	Haas
5,737,416 A	4/1998	Cooper
5,870,543 A	2/1999	Ronning
5,883,954 A	3/1999	Ronning

(57) **ABSTRACT**

The invention is an apparatus that facilitates access to encrypted digital media to accept verification and authentication from an excelsior enabler using at least one token and at least one electronic identification. The at least one electronic identification could be a device serial number, a networking MAC address, or a membership ID reference from a web service. Access to the product is also managed with a plurality of secondary enablers using the at least one electronic identification reference.

**26 Claims, 7 Drawing Sheets**



## US 8,402,555 B2

Page 2

## U.S. PATENT DOCUMENTS

7,567,987 B2 7/2009 Shappell et al.  
 7,568,111 B2 7/2009 Alve  
 7,571,328 B2 8/2009 Baumert  
 7,594,275 B2 9/2009 Zhu  
 7,610,630 B2 10/2009 Ji  
 7,624,417 B2 11/2009 Dua  
 7,634,734 B2 12/2009 Fuller et al.  
 7,689,823 B2 3/2010 Shen  
 7,702,592 B2 4/2010 Taylor  
 8,250,145 B2 8/2012 Zuckerberg  
 8,280,959 B1 10/2012 Zuckerberg  
 2002/0010759 A1 1/2002 Hitson  
 2002/0157002 A1 10/2002 Messerges  
 2003/0018491 A1 1/2003 Nakahara  
 2003/0051149 A1 3/2003 Robert  
 2003/0220880 A1 11/2003 Lao  
 2004/0024670 A1 2/2004 Valenzuela  
 2004/0062400 A1 4/2004 Sovio  
 2004/0162786 A1 8/2004 Cross  
 2004/0220878 A1 11/2004 Lao  
 2005/0065891 A1 \* 3/2005 Lee et al. .... 705/59  
 2005/0066353 A1 3/2005 Fransdonk  
 2005/0138406 A1 6/2005 Cox  
 2005/0182727 A1 8/2005 Robert  
 2005/0182931 A1 8/2005 Robert  
 2005/0198510 A1 9/2005 Robert  
 2005/0216752 A1 9/2005 Robert  
 2006/0036554 A1 2/2006 Schrock  
 2006/0173787 A1 8/2006 Weber  
 2006/0173789 A1 8/2006 Baumert  
 2006/0259852 A1 11/2006 Upendran  
 2006/0259982 A1 11/2006 Upendran  
 2007/0010334 A1 1/2007 Multerer  
 2007/0055887 A1 3/2007 Cross  
 2007/0156719 A1 7/2007 Upendran  
 2007/0179854 A1 8/2007 Ziv  
 2007/0180485 A1 8/2007 Dua  
 2007/0250445 A1 10/2007 Ache  
 2007/0266095 A1 11/2007 Billsus et al.  
 2008/0010685 A1 \* 1/2008 Holtzman et al. .... 726/27  
 2008/0027869 A1 1/2008 Kalker  
 2008/0091606 A1 4/2008 Grecia  
 2008/0109911 A1 5/2008 Tedesco  
 2008/0114992 A1 5/2008 Robert  
 2008/0137869 A1 6/2008 Robert  
 2008/0165956 A1 7/2008 Zhu  
 2008/0209576 A1 8/2008 Nooning  
 2009/0012805 A1 1/2009 Schnell  
 2009/0049556 A1 2/2009 Vrielink  
 2009/0083541 A1 \* 3/2009 Levine ..... 713/165  
 2009/0086975 A1 4/2009 Robert  
 2009/0089884 A1 4/2009 Robert  
 2009/0100060 A1 4/2009 Livnat et al.  
 2009/0106850 A1 4/2009 Robert  
 2009/0164776 A1 6/2009 Tuoriniemi  
 2009/0183010 A1 7/2009 Schnell  
 2009/0217036 A1 8/2009 Irwin  
 2009/0254930 A1 10/2009 Lo  
 2009/0257591 A1 10/2009 Mithal  
 2009/0265278 A1 10/2009 Wang  
 2009/0299963 A1 12/2009 Pippuri  
 2009/0307078 A1 12/2009 Mithal  
 2009/0327702 A1 12/2009 Schnell  
 2009/0328228 A1 12/2009 Schnell  
 2010/0027796 A1 2/2010 Robert  
 2010/0043077 A1 2/2010 Robert  
 2010/0057527 A1 3/2010 Robert  
 2010/0057871 A1 3/2010 Kaplan  
 2010/0100899 A1 \* 4/2010 Bradbury et al. .... 725/29  
 2010/0299264 A1 11/2010 Berger et al.  
 2011/0145896 A1 6/2011 Berger  
 2011/0208695 A1 8/2011 Anand  
 2011/0265157 A1 10/2011 Ryder  
 2011/0288946 A1 \* 11/2011 Baiya et al. .... 705/26.1  
 2011/0313898 A1 12/2011 Singhal  
 2011/0320345 A1 12/2011 Taveau  
 2012/0030291 A1 2/2012 Silver  
 2012/0036041 A1 2/2012 Hesselink

2012/0041829 A1 2/2012 Rothschild  
 2012/0066052 A1 3/2012 Robert  
 2012/0079095 A1 3/2012 Evans  
 2012/0079126 A1 3/2012 Evans  
 2012/0079276 A1 3/2012 Evans  
 2012/0079606 A1 3/2012 Evans  
 2012/0095871 A1 4/2012 Dorsey  
 2012/0095906 A1 4/2012 Dorsey  
 2012/0095916 A1 4/2012 Dorsey  
 2012/0124610 A1 5/2012 Shintani  
 2012/0124611 A1 5/2012 Shintani  
 2012/0124612 A1 5/2012 Adimatyam  
 2012/0124613 A1 5/2012 Reddy  
 2012/0124614 A1 5/2012 Shintani  
 2012/0124678 A1 5/2012 Shintani  
 2012/0130903 A1 5/2012 Dorsey  
 2012/0150727 A1 6/2012 Nuzzi  
 2012/0166333 A1 6/2012 von Behren  
 2012/0173333 A1 7/2012 Berger  
 2012/0173431 A1 7/2012 Ritchie  
 2012/0173625 A1 7/2012 Berger  
 2012/0191553 A1 7/2012 Sathe  
 2012/0254340 A1 10/2012 Velummylum  
 2012/0255033 A1 10/2012 Dwivedi  
 2012/0290376 A1 11/2012 Dryer  
 2012/0296741 A1 11/2012 Dykes  
 2012/0310828 A1 12/2012 Hu  
 2013/0007892 A1 1/2013 Inooka

## FOREIGN PATENT DOCUMENTS

JP 2007-183935 A 7/2007  
 KR 10-2004-0107602 A 12/2004  
 KR 10-2005-0028244 A 3/2005  
 KR 10-2005-0060685 A 6/2005  
 KR 10-0708203 B1 4/2007  
 WO 2008/111052 A2 9/2008

## OTHER PUBLICATIONS

Simon L. Garfinkel, "Email-Based Identification and Authentication: An Alternative to PKI?", IEEE Security & Privacy, <http://computer.org/security/>, published Nov. 2003, pp. 20-26.  
 Video Interview—Title: Mitch Singer, Sony Pictures—Publication Source: Youtube.com [URL: <http://youtu.be/nqISakADFII>]—(Internet Publication Jun. 24, 2008) Note: Attached URL for Media NPL Reference.  
 Video Interview—Title: Jeff Bewkes and Brian Roberts discuss the TV Everywhere model and upcoming trial on Comcast—Publication Source: Youtube.com [URL: <http://youtu.be/q8Rt9idJV9I>]—(Internet Publication Jun. 25, 2009) Note: Attached URL for Media NPL Reference.  
 Author: William Grecia (patent applicant)—Publication Source: Amazon Web Services Products and Solutions Catalog—Title: STR3EM Digital Distribution System (Ultraviolet—Keychest)—Internet Publication: <http://aws.amazon.com/customerapps/2621> [Publication date: Jun. 22, 2009].  
 Author: Nicholas Deleon—Publication Source: TechCrunch—Title: Movie studios launch Epix: 720p streaming video for free—Internet Publication: <http://techcrunch.com/2009/06/08/movie-studios-launch-epix-720p-streaming-video-for-free/> [Publication date: Jun. 8, 2009].  
 Author: Matt Burns—Publication Source: TechCrunch—Title: TV Everywhere is Comcast and Time Warner's answer to free Internet video—Internet Publication: <http://techcrunch.com/2009/06/24/tv-everywhere-is-comcast-and-time-warners-answer-to-free-internet-video/> [Publication date: Jun. 24, 2009].  
 Author—Michael Arrington—Movie Labels to Launch New "Open Market" Play Anywhere Scheme As Last Ditch Effort to Save DRM—Publication Source: TechCrunch.com [URL: <http://techcrunch.com/2008/08/26/movie-labels-to-launch-new-open-market-play-anywhere-scheme-as-last-ditch-effort-to-save-drm/>]—(Internet Publication Aug. 26, 2008).  
 Author—Mitch Singer—Developing the Digital Market—Publication Source: TechCrunch.com [URL: <http://tctechcrunch.files.wordpress.com/2008/08/singer.pdf>]—(Internet Publication Aug. 26, 2008).

## US 8,402,555 B2

Page 3

Author: Brian Fox—Title: DECE UltraViolet Response to STR3EM Licensing Offer—Internet Publication: <http://www.docstoc.com/docs/100643534/DECE-Note?> [Publication date: Oct. 25, 2011]. Abandoned USPTO U.S. Appl. No. 61/303,292 (evidence to overcome examiner's last rejection in accordance with MPEP 2142).

Author—Preston Gralla—Digital River Launches DRM Solution for Software Publishers—Publication Source: [informationweek.com](http://www.informationweek.com/news/18901739) [URL: <http://www.informationweek.com/news/18901739>]—(Internet Publication Date: Apr. 15, 2004).

Author—Digital River Corporation—Digital River Announces New Digital Rights Management Service—Publication Source: [digitalriver.com](http://www.digitalriver.com/corporate/press_releases/pr_328.shtml) [URL: [http://www.digitalriver.com/corporate/press\\_releases/pr\\_328.shtml](http://www.digitalriver.com/corporate/press_releases/pr_328.shtml)]—(Internet Publication Date: Jul. 14, 2003).

Author—Digital River Corporation—Digital River SoftwarePassport Copyright software—Publication Source: [siliconrealms.com](http://www.siliconrealms.com/) [URL: <http://www.siliconrealms.com/>]—(Internet Publication), Aug. 26, 2010.

Internet publication: Nook Color LendMe [www.barnesandnoble.com/](http://www.barnesandnoble.com/), Jan. 21, 2011.

Internet publication: Coral consortium “Scenario” [www.coral-interop.org](http://www.coral-interop.org), Feb. 27, 2006.

Author—wikipedia.org—Steam (software)—Publication Source: [wikipedia.org](http://en.wikipedia.org/wiki/Steam_(software)) [URL: [http://en.wikipedia.org/wiki/Steam\\_\(software\)](http://en.wikipedia.org/wiki/Steam_(software))], Aug. 13, 2004.

Author—William Grecia—STR3EM Windows Java C++ written code copyright and support documentation—Publication Source: [str3em.com](http://www.str3em.com) [URL: <http://www.str3em.com>]—(Software Copyright Publication Date and Invention Reduced to Practice: Sep. 3, 2009).

Author—William Grecia—Next Generation Digital Delivery STR3EM Ecosystem Replaces DVD and Blu-Ray—Publication Source: [mi2n.com](http://mi2n.com) [URL: [http://mi2n.com/press.php3?press\\_nb=130517](http://mi2n.com/press.php3?press_nb=130517)]—(Internet Publication Date: May 28, 2010).

Author—Facebook Corporation—Graph API documentation—Publication Source: [facebook.com](http://developers.facebook.com/docs/api) [URL: <http://developers.facebook.com/docs/api>]—(Internet Publication Update: Apr. 21, 2010).

Author—Amazon Inc—Amazon Web Services API documentation—Publication Source: [aws.amazon.com](http://aws.amazon.com) [URL: <http://aws.amazon.com>]—(Internet Publication), Jul. 16, 2002.

Author—Rick Merritt—Analysis: Hollywood's next digital media gambit—Publication Source: [eetimes.com](http://www.eetimes.com) [URL: <http://www.eetimes.com/design/audio-design/4005862/Analysis-Hollywood-s-next-digital-media-gambit>]—(Internet Publication Date: Nov. 2, 2008).

Author—Ethan Smith—Disney Touts a Way to Ditch the DVD—Publication Source: Wall Street Journal Online [URL: [http://online.wsj.com/article/NA\\_WSJ\\_PUB:SB10001424052748703816204575585650026945222.html](http://online.wsj.com/article/NA_WSJ_PUB:SB10001424052748703816204575585650026945222.html)]—(Internet Publication Date: Sep. 21, 2009).

Author—Neda Ulaby—Introducing UltraViolet: Buy Your Digital Movie Once, Play It Anywhere?—Publication Source: NPR Online [URL: <http://www.npr.org/blogs/monkeysee/2010/07/19/128626624/introducing-ultraviolet-buy-your-movie-once-play-it-anywhere>]—(Internet Publication Date: Jul. 20, 2010).

Author—William Grecia—The Retail Zip Company Releases Secure Electronic Media Format STR3EM to Replace DVD and Blu-ray—Publication Source: [mi2n.com](http://mi2n.com) [URL: [http://mi2n.com/press.php3?press\\_nb=1122843](http://mi2n.com/press.php3?press_nb=1122843)]—(Internet Publication Date: Sep. 3, 2009).

Author—Apple Corporation—iTunes application copyright—Publication Source: [apple.com](http://apple.com) [URL: [www.apple.com/itunes](http://www.apple.com/itunes)]—(Copyright Publication Date: Jan. 9, 2011).

Author—Microsoft Corporation—Zune application copyright—Publication Source: [zune.net](http://zune.net) [URL: [www.zune.net](http://www.zune.net)]—(Copyright Publication Date: Nov. 14, 2006).

Author—Netflix Corporation—Netflix application copyright—Publication Source: [netflix.com](http://netflix.com) [URL: [www.netflix.com](http://www.netflix.com)]—(Copyright Publication), Jan. 15, 2007.

Author—Best Buy Corporation—CinemaNow application copyright—Publication Source: [cinemanow.com](http://cinemanow.com) [URL: [www.cinemanow.com/](http://www.cinemanow.com/)]—(Copyright Publication), May 18, 2010.

Author—Best Buy Corporation—Napster application copyright—Publication Source: [napster.com](http://napster.com) [URL: [www.napster.com/](http://www.napster.com/)]—(Copyright Publication), May 1, 2006.

Author—Google Corporation—YouTube application copyright—Publication Source: [youtube.com](http://youtube.com) [URL: [www.youtube.com/](http://www.youtube.com/)]—(Copyright Publication), Feb. 14, 2005.

Author—Wal-Mart Corporation—Vudu application copyright—Publication Source: [vudu.com](http://vudu.com) [URL: [www.vudu.com/](http://www.vudu.com/)]—(Copyright Publication), Sep. 6, 2007.

Author—Connected Media Experience Org—CMX specification—Publication Source: [connectedmediaexperience.org](http://connectedmediaexperience.org) [URL: [www.connectedmediaexperience.org/technicaloverview.html](http://www.connectedmediaexperience.org/technicaloverview.html)]—(Internet Publication), Jan. 27, 2009.

Author—SMPTE Org—Digital Cinema DCP MXF specifications—Publication Source: [smpte.org](http://smpte.org) [URL: [www.smpte.org/standards](http://www.smpte.org/standards)]—(Internet Publication), Oct. 8, 2002.

Author—Wikipedia Org—Xbox Live Marketplace and Zune Marketplace—Publication Source: [wikipedia.org](http://en.wikipedia.org/wiki/Xbox_Live_Marketplace_and_Zune_Marketplace) [URL: [http://en.wikipedia.org/wiki/Xbox\\_Live\\_Marketplace\\_and\\_Zune\\_Marketplace](http://en.wikipedia.org/wiki/Xbox_Live_Marketplace_and_Zune_Marketplace)]—(Internet Publication), Jan. 18, 2006.

Author—Dan Franks—First Look. iTunes Digital Copy—Publication Source: [macworld.com](http://macworld.com) [URL: [www.macworld.com/article/131751/2008/01/digitalcopy.html](http://www.macworld.com/article/131751/2008/01/digitalcopy.html)]—(Internet Publication Jan. 22, 2008).

Author—Rich Fiscus—Review—Is DVD Digital Copy worth the trouble?—Publication Source: [afterdawn.com](http://afterdawn.com) [URL: [www.afterdawn.com/news/article.cfm/2009/11/18/review\\_is\\_dvd\\_digital\\_copy\\_worth\\_the\\_trouble](http://www.afterdawn.com/news/article.cfm/2009/11/18/review_is_dvd_digital_copy_worth_the_trouble)]—(Internet Publication Nov. 18, 2009).

Author—Wikipedia Org—Digital rights management—Publication Source: [wikipedia.org](http://en.wikipedia.org/wiki/Digital_Rights_Management) [URL: [http://en.wikipedia.org/wiki/Digital\\_Rights\\_Management](http://en.wikipedia.org/wiki/Digital_Rights_Management)]—(Internet Publication), Sep. 22, 2002.

Author—Wikipedia Org—Application programming interface—Publication Source: [wikipedia.org](http://en.wikipedia.org/wiki/Api) [URL: <http://en.wikipedia.org/wiki/Api>]—(Internet Publication), Jul. 30, 2001.

Author—Wikipedia Org—Steam (content delivery)—Publication Source: [wikipedia.org](http://en.wikipedia.org/wiki/Steam_(content_delivery)) [URL: [http://en.wikipedia.org/wiki/Steam\\_\(content\\_delivery\)](http://en.wikipedia.org/wiki/Steam_(content_delivery))]—(Internet Publication), Sep. 13, 2004.

Author—Ben Drawbaugh—Disney's KeyChest is not DRM—Publication Source: [engadget.com](http://engadget.com) [URL: [www.engadget.com/2010/01/10/disneys-keychest-is-not-drm](http://www.engadget.com/2010/01/10/disneys-keychest-is-not-drm)]—(Internet Publication Jan. 10, 2010).

Author—Richard Lawler—DECE & Keychest both laying claim to friendly DRM of the future title—Publication Source: [engadget.com](http://engadget.com) [URL: [www.engadget.com/2010/01/06/dece-and-keychain-both-laying-claim-to-friendly-drm-of-the-future](http://www.engadget.com/2010/01/06/dece-and-keychain-both-laying-claim-to-friendly-drm-of-the-future)]—(Internet Publication Jan. 6, 2010).

\* cited by examiner



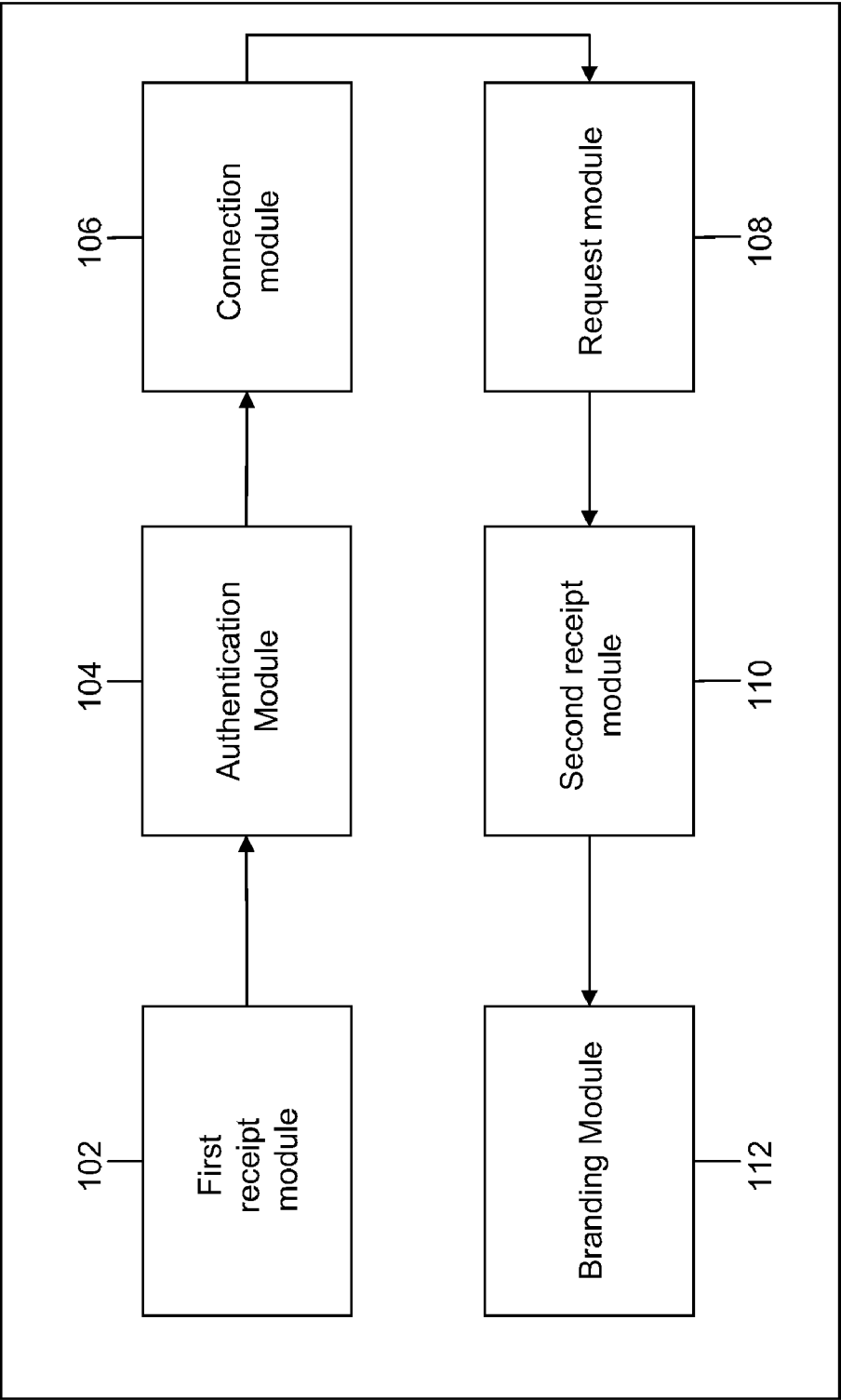


FIG.1

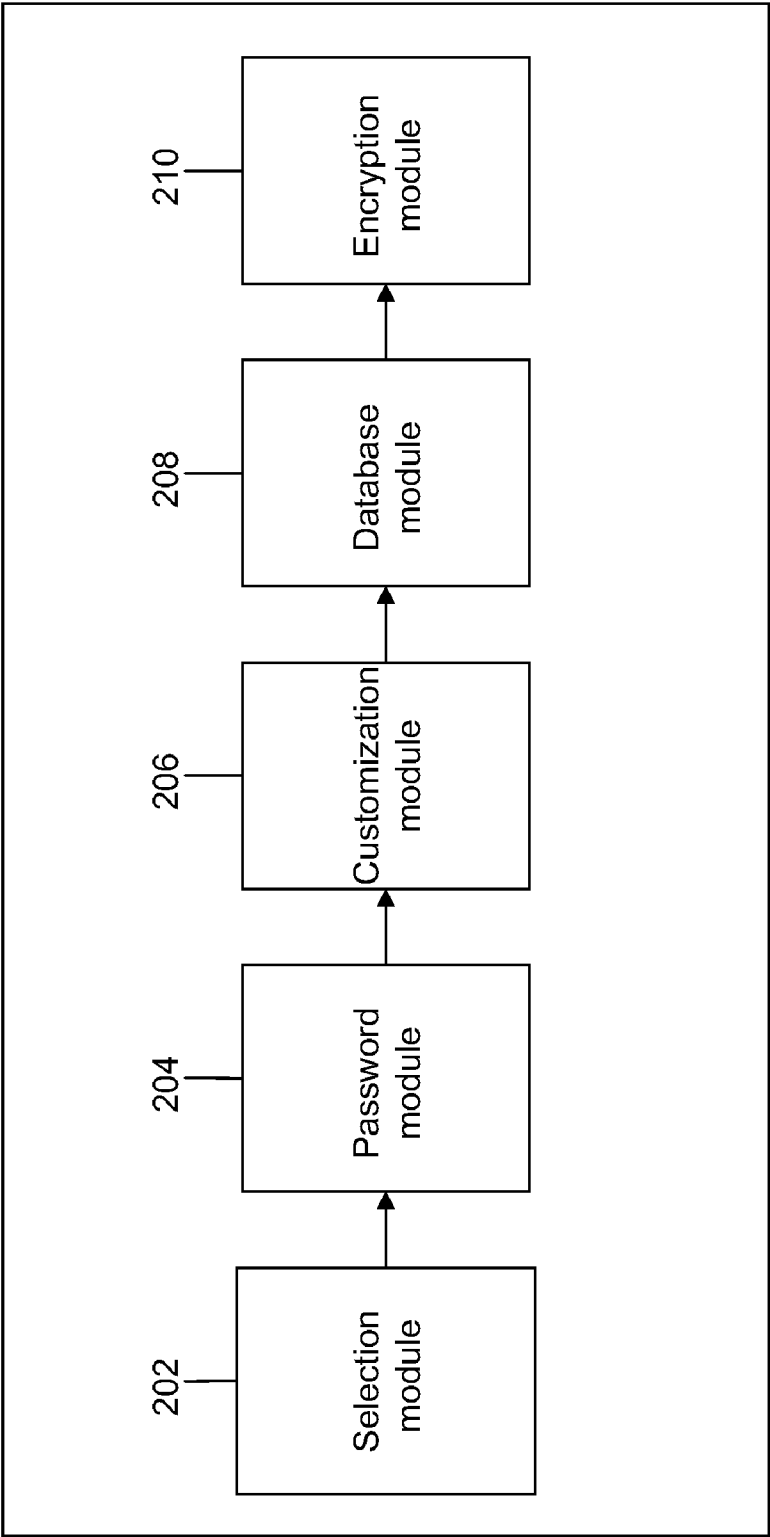


FIG.2



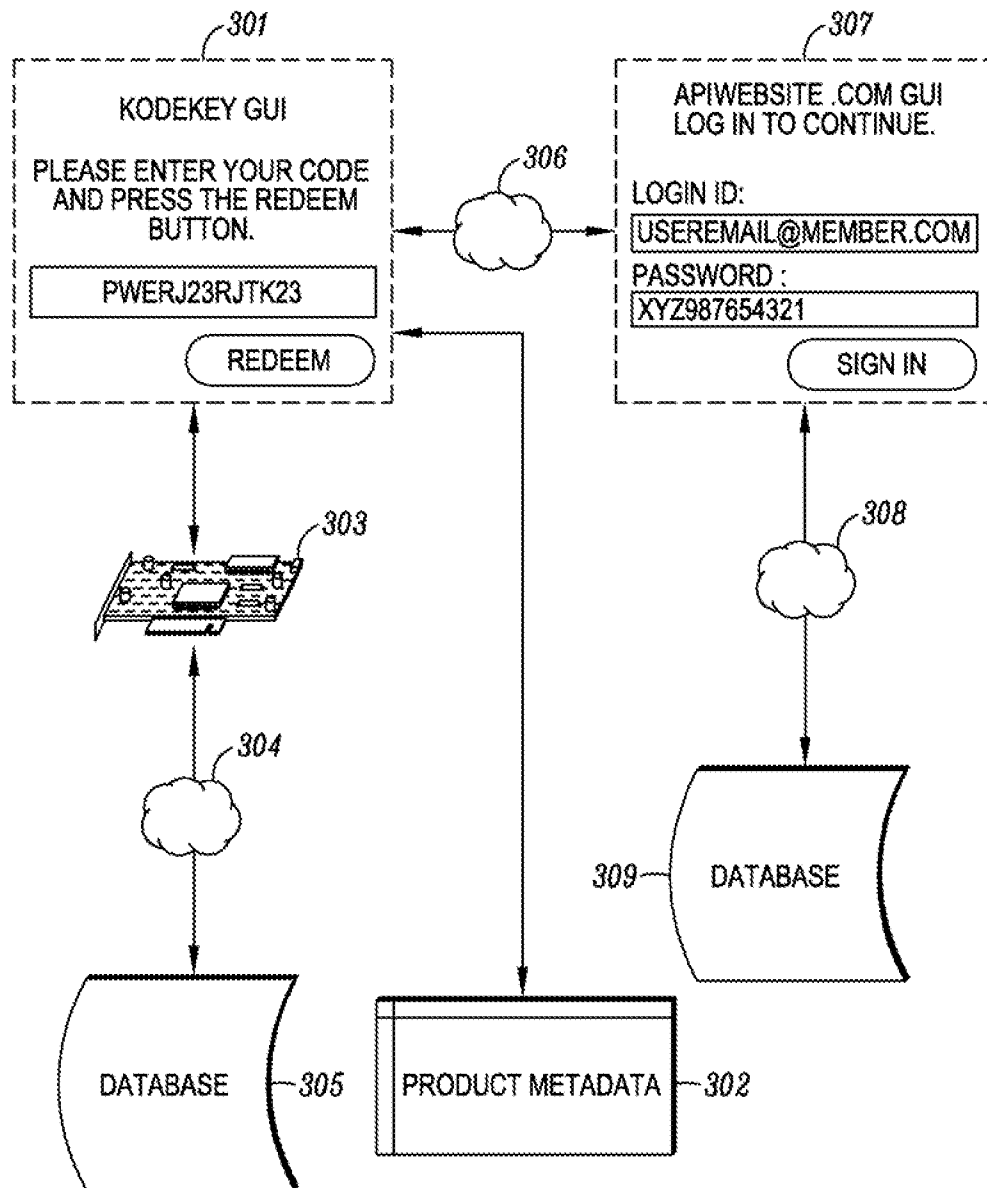


FIG. 3

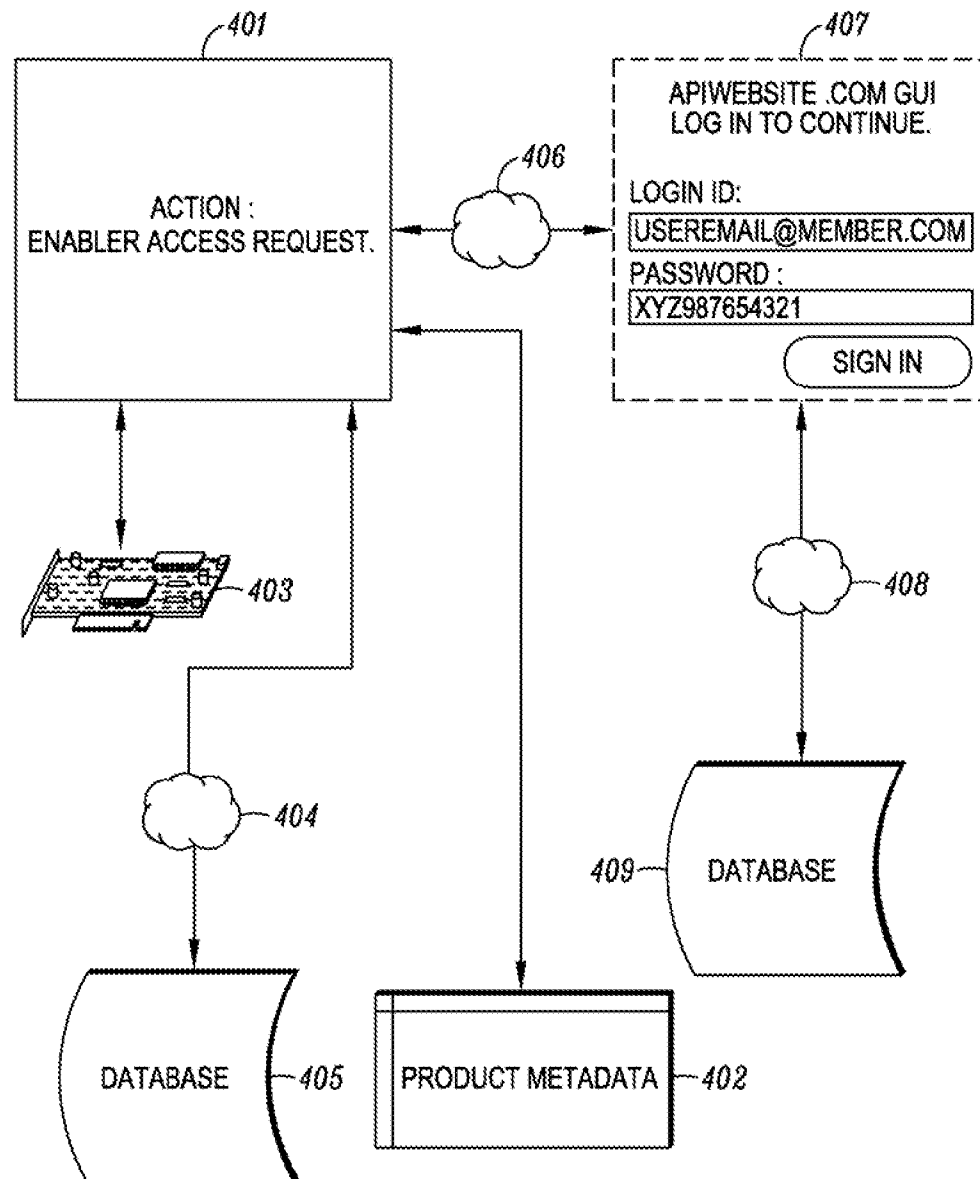


FIG. 4

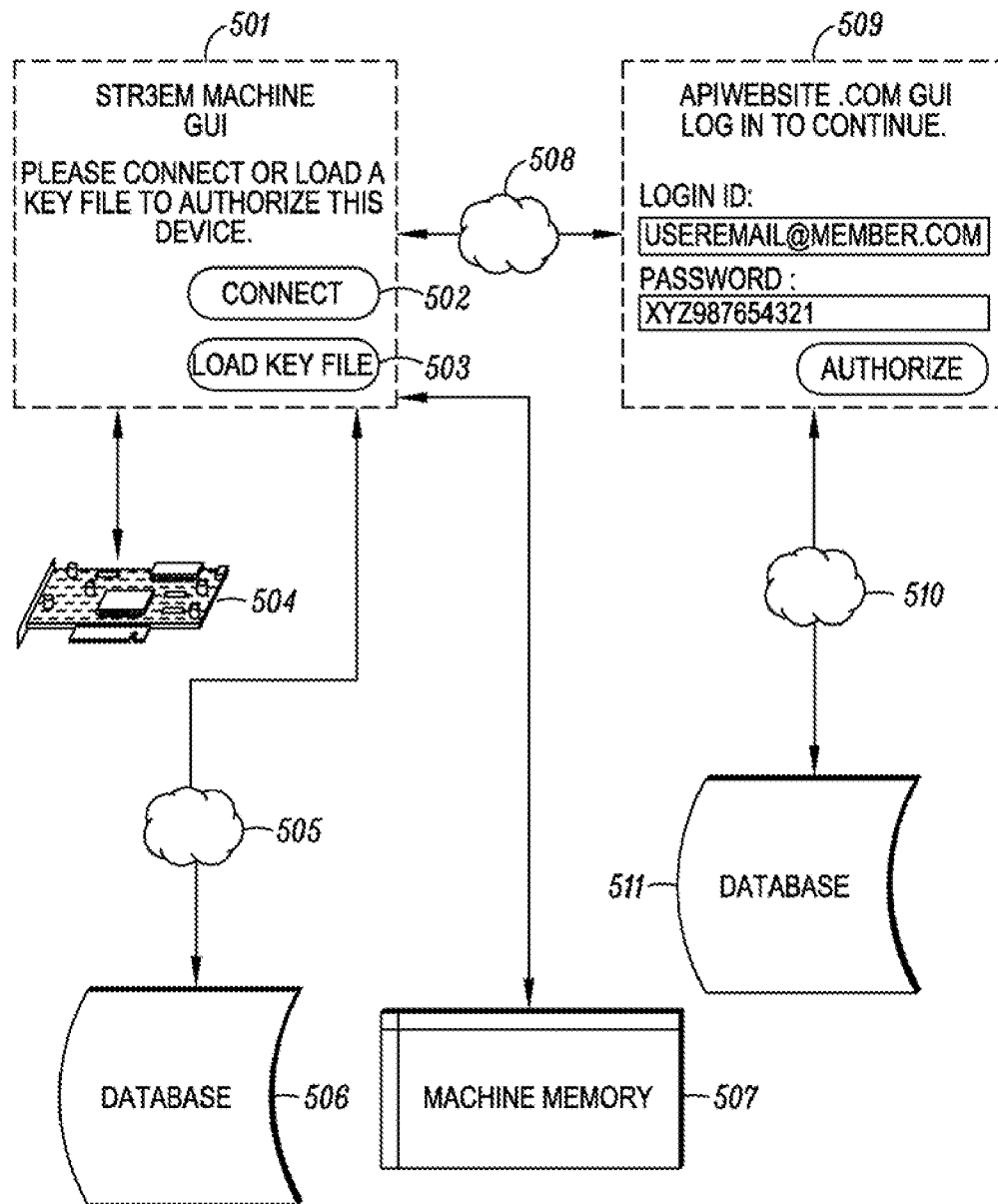


FIG. 5

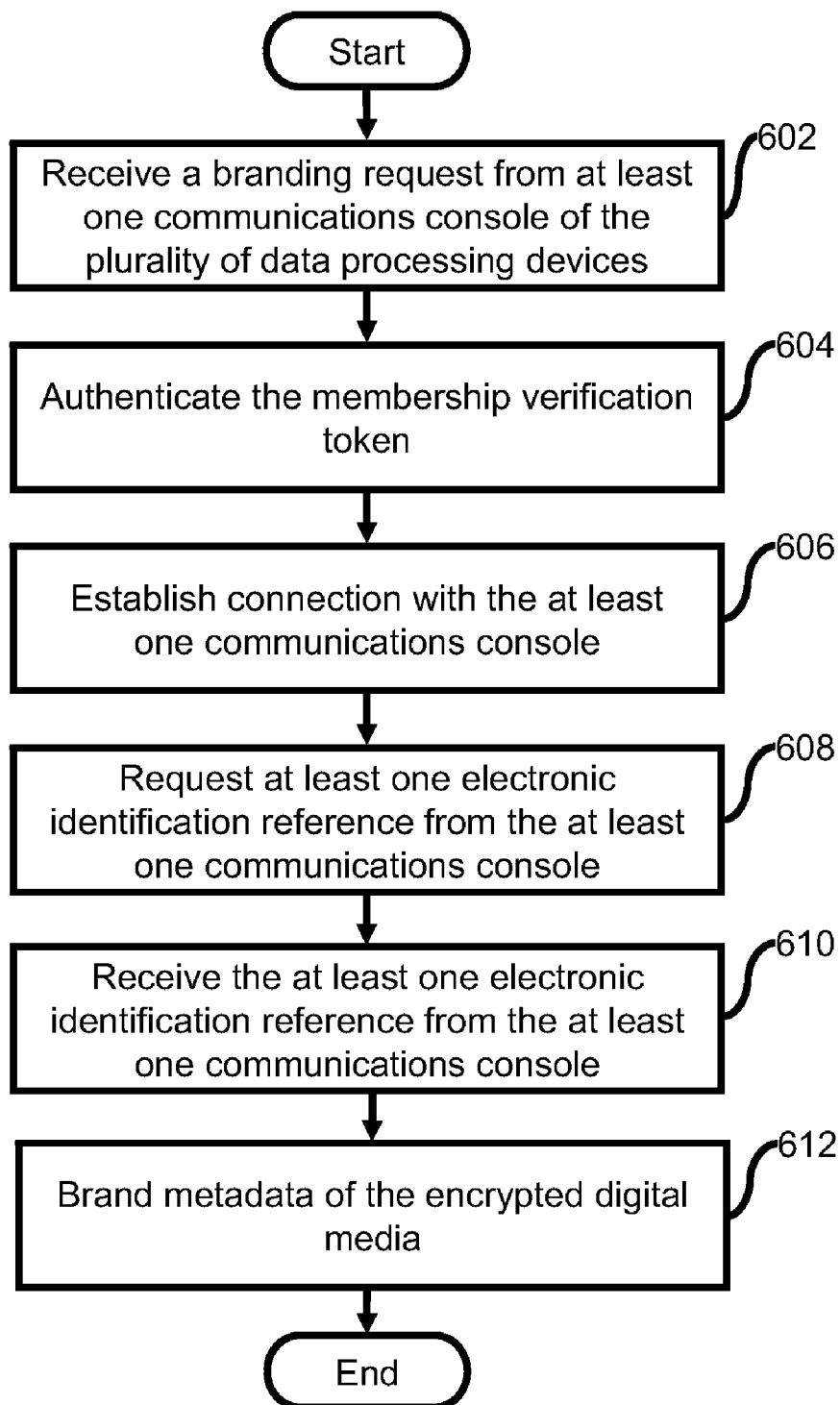


FIG.6

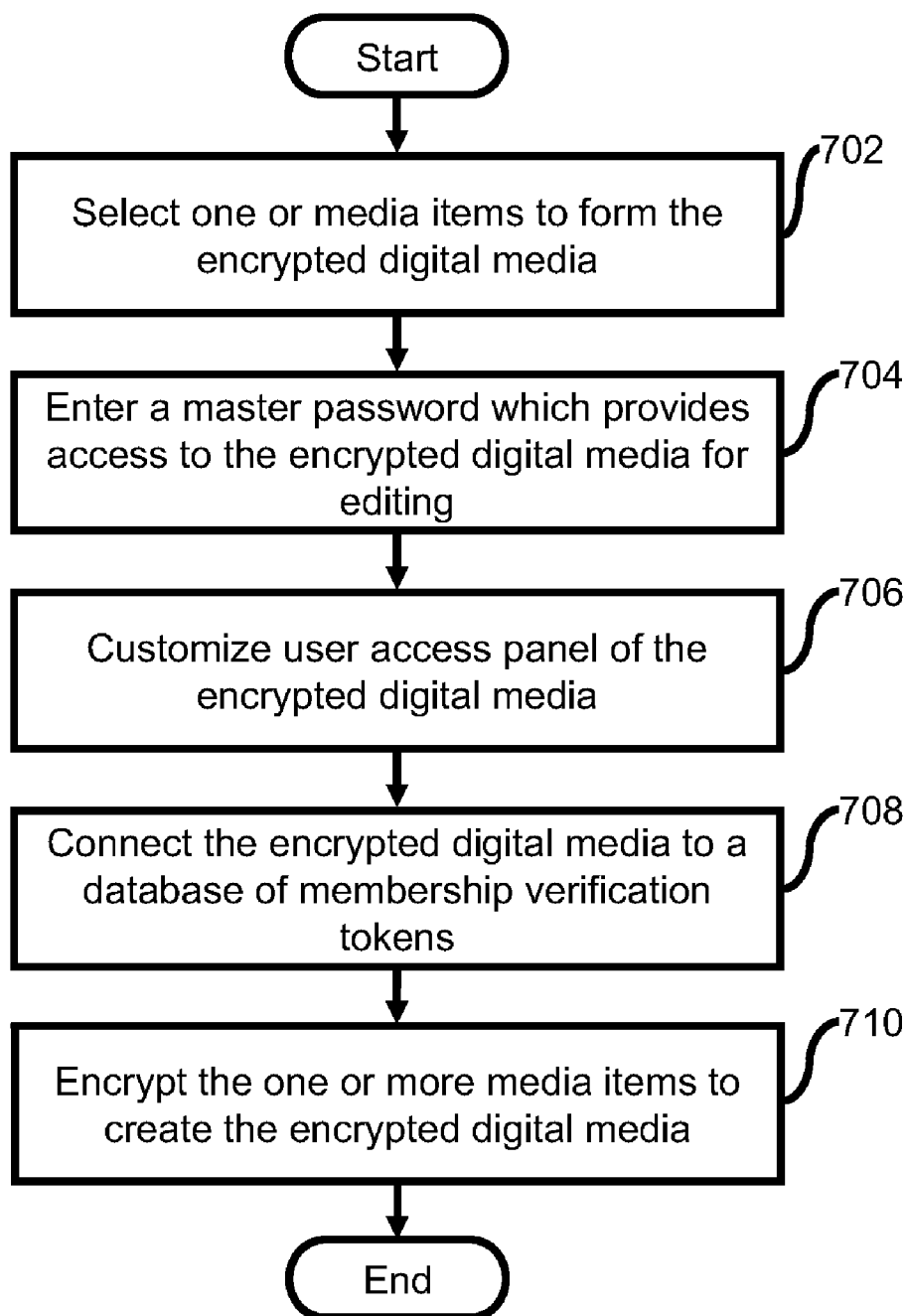


FIG.7

US 8,402,555 B2

1

**PERSONALIZED DIGITAL MEDIA ACCESS  
SYSTEM (PDMAS)****CROSS-REFERENCE TO RELATED  
APPLICATIONS**

This application is a continuation of, and claims the priority benefit of, U.S. patent application co-pending Ser. No. 12/985,351 titled PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS) filed Jan. 6, 2011; which is a continuation of, and claims the priority benefit of, U.S. patent application Ser. No. 12/728,218 filed Mar. 21, 2010 now abandoned, which are both incorporated herein by reference, in their entirety.

**BACKGROUND OF THE INVENTION****1. Field of the Invention**

The present invention relates to the field of digital rights management schemes used by creators of electronic products to protect commercial intellectual property copyrights privy to illegal copying using computerized devices. More specifically, the present invention teaches a more personal system of digital rights management which employs electronic ID, as part of a web service membership, to manage access rights across a plurality of devices.

**2. Description of the Prior Art**

Digital rights management (DRM) is a generic term for access control technologies used by hardware manufacturers, publishers, copyright holders and individuals to impose limitations on the usage of digital content across devices. DRM refers to any technology that inhibits undesirable or illegal uses of the digital content. The term generally doesn't refer to forms of copy protection that can be circumvented without modifying the file or device, such as serial numbers or key files. It can also refer to restrictions associated with specific instances of digital works or devices.

Traditional DRM schemes are defined as authentication components added to digital files that have been encrypted from public access. Encryption schemes are not DRM methods but DRM systems are implemented to use an additional layer of authentication in which permission is granted for access to the cipher key required to decrypt files for access. A computer server is established to host decryption keys and to accept authentication keys from Internet connected client computers running client software in which handles the encrypted files. The server can administer different authorization keys back to the client computer that can grant different sets of rules and a time frame granted before the client is required to connect with the server to reauthorize access permissions. In some cases content can terminate access after a set amount of time, or the process can break if the provider of the DRM server ever ceases to offer services.

In the present scenario, consumer entertainment industries are in the transition of delivering products on physical media such as CD and DVD to Internet delivered systems. The Compact Disc, introduced to the public in 1982, was initially designed as a proprietary system offering strict media to player compatibility. As the popularity of home computers and CD-ROM drives rose, so did the availability of CD ripping applications to make local copies of music to be enjoyed without the use of the disc. After a while, users found ways to share digital versions of music in the form of MP3 files that could be easily shared with family and friends over the Internet. The DVD format introduced in 1997 included a new apparatus for optical discs technology with embedded copy protection schemes also recognized as an early form of DRM.

2

With internet delivered music and video files, DRM schemes has been developed to lock acquired media to specific machines and most times limiting playback rights to a single machine or among a limited number of multiple machines regardless of the model number. This was achieved by writing the machine device ID to the metadata of the media file, then cross referencing with a trusted clearinghouse according to pre-set rules. DRM systems employed by DVD and CD technologies consisted of scrambling (also known as encryption) disc sectors in a pattern to which hardware developed to unscramble (also known as decryption) the disc sectors are required for playback. DRM systems built into operating systems such as Microsoft Windows Vista block viewing of media when an unsigned software application is running to prevent unauthorized copying of a media asset during playback. DRM used in computer games such as SecuROM and Steam are used to limit the amount of times a user can install a game on a machine. DRM schemes for e-books include embedding credit card information and other personal information inside the metadata area of a delivered file format and restricting the compatibility of the file with a limited number of reader devices and computer applications.

In a typical DRM system, a product is encrypted using Symmetric block ciphers such as DES and AES to provide high levels of security. Ciphers known as asymmetric or public key/private key systems are used to manage access to encrypted products. In asymmetric systems the key used to encrypt a product is not the same as that used to decrypt it. If a product has been encrypted using one key of a pair it cannot be decrypted even by someone else who has that key. Only the matching key of the pair can be used for decryption. After receiving an authorization token from a first-use action are usually triggers to decrypt block ciphers in most DRM systems. User rights and restrictions are established during this first-use action with the corresponding hosting device of a DRM protected product.

Examples of such prior DRM art include Hurtado (U.S. Pat. No. 6,611,812) who described a digital rights management system, where upon request to access digital content, encryption and decryption keys are exchanged and managed via an authenticity clearing house. Other examples include Alve (U.S. Pat. No. 7,568,111) who teaches a DRM and Tuoriniemi (U.S. Pat. No. 20090164776) who described a management scheme to control access to electronic content by recording use across a plurality of trustworthy devices that has been granted permission to work within the scheme.

Recently, DRM schemes have proven unpopular with consumers and rights organizations that oppose the complications with compatibility across machines manufactured by different companies. Reasons given to DRM opposition range from limited device playback restrictions to the loss of fair-use which defines the freedom to share media products will family members.

Prior art DRM methods rely on content providers to maintain computer servers to receive and send session authorization keys to client computers with an Internet connection. Usually rights are given from the server for an amount of time or amount of access actions before a requirement to reconnect with the server is required for reauthorization. At times, content providers will discontinue servers or even go out of business some years after DRM encrypted content was sold to consumers causing the ability to access files to terminate.

In the light of the foregoing discussion, the current states of DRM measures are not satisfactory because unavoidable issues can arise such as hardware failure or property theft that could lead to a paying customer loosing the right to recover purchased products. The current metadata writable DRM

US 8,402,555 B2

3

measures do not offer a way to provide unlimited interoperability between different machines. Therefore, a solution is needed to give consumers the unlimited interoperability between devices and “fair use” sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments.

#### SUMMARY OF THE INVENTION

An object of the present invention is to provide unlimited interoperability of digital media between unlimited machines with management of end-user access to the digital media.

In accordance with an embodiment of the present invention, the invention is a process of an apparatus which in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods (herein referred to as The App) is used to: handle at least one branding action which could include post read and write requests of at least one writable metadata as part of at least one digital media asset to identify and manage requests from at least one excelsior enabler, and can further identify and manage requests from a plurality of connected second enablers; with at least one token and at least one electronic identification reference received from the at least one excelsior enabler utilizing at least one membership. Here, controlled by the at least one excelsior enabler, The App will proceed to receive the at least one token to verify the authenticity of the branding action and further requests; then establish at least one connection with at least one programmable communications console of the at least one membership to request and receive the at least one electronic identification reference; and could request and receive other data information from the at least one membership. The method then involves sending and receiving variable data information from The App to the at least one membership to verify a preexisting the at least one branding action of the at least one writable metadata as part of the at least one digital media asset; or to establish permission or denial to execute the at least one branding action or the post read and write requests of the at least one writable metadata. To do this, controlled by the at least one excelsior enabler. The App may establish at least one connection, which is usually through the Internet, with a programmable communications console, which is usually a combination of an API protocol and graphic user interface (GUI) as part of a web service. In addition, the at least one excelsior enabler provides reestablished credentials to the programmable communications console as part of the at least one membership, in which The App is facilitating and monitoring, to authenticate the data communications session used to send and receive data requests between the at least one membership and The App.

In accordance with another embodiment of the present invention, the present invention teaches a method for monitoring access to an encrypted digital media and facilitating unlimited interoperability between a plurality of data processing devices. The method comprises receiving a branding request from at least one communications console of the plurality of data processing devices, the branding request being a read and write request of metadata of the encrypted digital media, the request comprising a membership verification token corresponding to the encrypted digital media. Subsequently, the membership verification token is authenticated, the authentication being performed in connection with a token database. Thereafter, connection with the at least one communications console is established. Afterwards, at least one electronic identification reference is requested from the at least one communications console. Further, the at least one

4

electronic identification reference is received from the at least one communications console. Finally, branding metadata of the encrypted digital media is performed by writing the membership verification token and the electronic identification reference into the metadata.

The present invention is particularly useful for giving users the freedom to use products outside of the device in which the product was acquired and extend unlimited interoperability with other compatible devices.

#### BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, the needs satisfied thereby, and the objects, features, and advantages thereof, reference now is made to the following description taken in connection with the accompanying drawings.

FIG. 1 shows a system for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 2 shows a system for authoring an encrypted digital media according to an embodiment of the present invention.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention.

Skilled artisans will appreciate that elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale. For example, the dimensions of some of the elements in the figures may be exaggerated relative to other elements to help to improve understanding of embodiments of the present invention.

#### DETAILED DESCRIPTION OF THE DRAWINGS

Before describing in detail the particular system and method for personalised digital media access system in accordance with an embodiment of the present invention, it should be observed that the present invention resides primarily in combinations of system components related to the device of the present invention.

Accordingly, the system components have been represented where appropriate by conventional symbols in the drawings, showing only those specific details that are pertinent to understanding the present invention so as not to obscure the disclosure with details that will be readily apparent to those of ordinary skill in the art having the benefit of the description herein.

In this document, relational terms such as ‘first’ and ‘second’, and the like may be used solely to distinguish one entity or action from another entity or action without necessarily requiring or implying any actual such relationship or order between such entities or actions. The terms ‘comprises’, ‘comprising’, or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that comprises a list of elements does not



## US 8,402,555 B2

5

include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. An element preceded by 'comprises . . . a' does not, without more constraints, preclude the existence of additional identical elements in the process, method, article, or apparatus that comprises the element.

The present invention is directed at providing infinite access rights of legally acquired at least one encrypted digital media asset to the content acquirer, explained in this document as the excelsior enabler, and optionally to their recognized friends and family, explained in this document as a plurality of secondary enablers. To explain further, the excelsior enabler and secondary enablers defined comprises human beings or computerized mechanisms programmed to process steps of the invention as would normally be done manually by a human being. Additionally, an apparatus used alone or in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods with a connection are needed (herein referred to as The App). To deliver the requirements of the invention, communicative and connected elements comprise: verification, authentication, electronic ID metadata branding, additional technical branding, and cross-referencing. The connection handling the communicative actions of the invention will usually be the Internet and can also be an internal apparatus cooperative. The App can further be defined as a Windows OS, Apple OS, Linux OS, and other operating systems hosting software running on a machine or device with a capable CPU, memory, and data storage. The App can be even further defined as a system on a chip (SOC), embedded silicon, flash memory, programmable circuits, cloud computing and runtimes, and other systems of automated processes.

The digital media assets used in this system are encrypted usually with an AES cipher and decryption keys are usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connection usually an Internet server. As explained earlier, the system we will discuss will work as a front-end to encrypted files as an authorization agent for decrypted access.

FIG. 1 shows a system **100** for monitoring access to an encrypted digital media according to an embodiment of the present invention. The system **100** includes a first recipient module **102**, an authentication module **104**, a connection module **106**, a request module **108**, a second receipt module **110** and a branding module **112**. The first receipt module **102** receives a branding request from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media and includes a membership verification token corresponding to the encrypted digital media. Examples of the encrypted digital media includes, and are not limited to, one or more of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

Subsequently, the authentication module **104** authenticates the membership verification token. The authentication is performed in connection with a token database. Further, the connection module **106** establishes communication with the at least one communication console.

According to an embodiment of the present invention, the connection is established through one of internet, intranet, Bluetooth, VPN, Infrared and LAN.

According to another embodiment of the present invention, the communication console is a combination of an Application Programmable interface (API) protocol and graphic user interface (GUI) as a part of web service. The API is a set of

6

routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services. The API is either one of language dependent or language independent.

The request module **108** requests at least one electronic identification reference from the at least one communication console. The second receipt module **110** receives the at least one electronic identification reference from the least one communication console. The branding module **112** brands metadata of the encrypted digital media by writing the membership verification token and the electronic identification into the metadata.

FIG. 2 shows a system **200** for authoring an encrypted digital media according to an embodiment of the present invention. The figure includes a selection module **202**, a password module **204**, a customization module **206**, a database module **208** and an encryption module **210**. The selection module **202** facilitates selection of one or more media items to form the encrypted digital media. Examples of the one or more media items include, and are not limited to, one or more of a video, an audio and a game.

According to an embodiment of the present invention, the one or more media items are one or more of remote URL links and local media files.

The password module **204** prompts the user to enter a master password which provides access to the encrypted digital media. Subsequently, the customization module **206** allows the user to customize the user access panel of the encrypted digital media.

According to an embodiment of the present invention, the customization module **206** facilitates adding one or more of a banner, a logo, an image, an advertisement, a tag line, a header message and textual information to the user access panel of the encrypted digital media.

Further, the database module **208** connects the encrypted digital media to a database of membership verification token required for decrypting the encrypted digital media.

According to an embodiment of the present invention, the membership verification token is a kodekey. The kodekey is a unique serial number assigned to the encrypted digital media.

The encryption module **210** encrypts the one or more media items to create the encrypted digital media.

According to an embodiment of the present invention, the system **200** further includes a watermark module. The watermark module watermarks information on the encrypted digital media, wherein the watermark is displayed during playback of the encrypted digital media.

According to another embodiment of the present invention, the system **200** further includes an access module. The access module allows the user to define access rights. Examples of the access rights include, but are not limited to, purchasing rights, rental rights and membership access rights.

According to yet another embodiment of the present invention, the system **200** further includes a name module. The name module allows the user to name the encrypted digital media.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention. The process is achieved by way of an enabler using an apparatus or otherwise known as an application in which facilitates digital media files. The apparatus interacts with all communicative parts required to fulfill the actions of the invention. The figure shows a Kodekey Graphical User Interface (GUI) **301**, a product metadata **302**, a networking card **303**, internet **304**, **306** and **308**, database **305** and **309** and an APIwebsite.com GUI **307**. A user posts a branding request via the Kodekey GUI interface **301**. The Kodekey GUI interface **301** is the GUI for entering token. The



US 8,402,555 B2

7

Kodekey GUI interface **301** prompts the user to enter the token and press the redeem button present on the Kodekey GUI interface **301**. The product metadata **302** is read/writable metadata associated with the digital media to be acquired. The networking card **303** facilitates querying of optional metadata branding process and referenced. The Kodekey GUI interface is connected to the database **305** via the internet **304** through the networking card **303**. The database **305** is the database used to read/write and store the tokens, also referred to as token database. The user is redirected to the APIwebsite.com GUI **307** through the internet **306**. The APIwebsite.com is the GUI to the membership API in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **307** prompts the user to enter a login id and a password to access the digital media which is acquired from the database **309** through the internet **308**. The database **309** is the database connected to the web service membership in which the user's electronic ID is queried from.

Examples of the encrypted digital files include, and are not limited to, a video file, an audio file, container formats, documents, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention. Subsequently, the communicative parts to cross-reference information stored in the metadata of the digital media asset are checked which has been previously handled by the process of FIG. 1. The figure shows an enabler access request **401**, a product metadata **402**, a networking card **403**, an internet **404**, **406** and **408**, a database **405** and **409** and an APIwebsite.com GUI **407**. The enabler access request **401** facilitates the user to make a request for the digital media. The product metadata **402** is read/writable metadata associated with the digital media to be acquired. The networking card **403** facilitates querying of optional metadata branding process and referenced. The database **405** is the database used to read/write and store the tokens. The APIwebsite.com GUI **407** is the GUI in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **407** prompts the user to enter a login id and a password to access the digital media from the database **409** through the internet **408**. The database **409** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory. The figure represents an authorization sequence action in which a machine is authorized to accept a personalized digital media file. The figure includes STR3EM Machine GUI **501** including the connect icon **502**, a load key file icon **503**, a networking card **504**, an internet **505**, **508** and **510**, a database **506** and **511**, a machine memory **507** and a APIwebsite.com GUI **509**. The STR3EM Machine GUI **501** prompts the user to connect or load a key file to authorize the device through the connect icon **502** and the load key file icon **503**. The STR3EM Machine GUI **501** is connected to the networking card **504**. The networking card **504** facilitates querying of optional metadata branding process and referenced. Further, the STR3EM machine GUI **501** is connected to the database **506** via the internet **505**. The database **506** is the database used to read/write and store the tokens. Moreover, STR3EM Machine GUI **501** is connected to the machine memory **507**. The machine memory **507** represents the internal memory of the machine or device so authorizations can be saved for access of the digital media. The API-

8

website.com GUI **509** is connected to the STR3EM machine GUI through the internet **508**. Further, APIwebsite.com GUI **509** is connected to the database **511** through the internet **510**. The APIwebsite.com GUI **509** prompts the user to enter the login id and a password to authorize the access to digital media. The database **511** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention. At step **602**, a branding request is made by a user from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media.

According to an embodiment of the present invention, the request includes a membership verification token corresponding to the encrypted digital media.

Subsequently, the membership verification token is authenticated at step **604**. The authentication is performed in connection with a token database. Further, connection with the at least one communication console is established at step **606**. Afterwards, at least one electronic identification reference is requested from the at least one communications console at the step **608**. At step **610**, at least one electronic identification reference is received from the at least one communication console. Finally, metadata of the encrypted digital media is branded by writing the membership verification token and the electronic identification reference into the metadata at the step **612**.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention. At step **702**, one or more media items are selected by the user to form the encrypted digital media. Subsequently, a master password is entered for providing access to the encrypted digital media for editing at step **704**. Afterwards, the user customizes the user panel of the encrypted digital media at step **706**. Further, the encrypted digital media is connected to a database of membership verification tokens required for decrypting the encrypted digital media at the step **708**. Finally, the one or more media items are encrypted to create the encrypted digital media at the step **710**.

According to various embodiments of the present invention, the verification is facilitated by at least one token handled by at least one excelsior enabler. Examples of the token include, and are not limited to, a structured or random password, e-mail address associated with an e-commerce payment system used to make an authorization payment, or other redeemable instruments of trade for access rights of digital media. Examples of e-commerce systems are PayPal, Amazon Payments, and other credit card services.

According to an embodiment of the present invention, an identifier for the digital media is stored in a database with another database of a list of associated tokens for cross-reference identification for verification.

According to an embodiment of the present invention, the database of a list of associated tokens includes Instant Payment Notification (IPN) received from successful financial e-commerce transactions that includes the identifier for the digital media; import of CSV password lists, and manually created reference phrases.

For this discussion, the structured or random password example will be used as reference. The structured or random passwords can be devised in encoded schemes to flag the apparatus of permission type such as: 1) Purchases can start a password sequence with "P" following a random number, so further example would be "PSJD42349MFJDF". 2) Rentals

US 8,402,555 B2

9

can start or end a password sequence with “R” plus (+) the number of days a rental is allowed, for example “R7” included in “R7SJDFHG**58473**” flagging a seven day rental. 3) Memberships can start or end a password sequence with “M” plus (+) optionally the length of months valid for example “M11DFJGH**34KF**” would flag an eleven-month membership period.

According to an embodiment of the present invention, the tokens are stored in a relational database such as MySQL or Oracle. Cloud storage systems such as Amazon’s Web Services Simple Storage Solution, or also known as S3, provides a highly available worldwide replicated infrastructure. In addition to S3, monetization offerings such as DevPay offer developers the opportunity to make money from applications developed to use the services.

The verification will reference to the S3 and DevPay services for example purposes only as many options such as FTP, SimpleDB, solid state storage and others can be used to host the token hosting needed for the verification element of this invention. The token represents permission from the content provider to grant access rights to the excelsior enabler and thereafter the plurality of secondary enablers. To set up the verification the content provider can manually or automatically generate a single or a plurality of structured or random password in which will represent the token. By using public or private access of S3 as part of an apparatus, the content provider can create empty text files giving each the name of the passwords generated. Because S3 is associated with a highly available worldwide infrastructure, to check this password token can be done my simply constructing a HTTP request from the apparatus and triggering follow up actions based on either a 200 HTTP response, which means OK at which point the next action can happen, or a 400 HTTP response which means ERROR at which point the verification process is voided. An additional token can be used to provide a flag to the apparatus that the verification element has been fulfilled for an initial verification token. Creating an alternate version of the first token by appending a reference to the end, for example, does this: “M11DFJGH34KF\_usergstr3em.com\_01\_01\_11”. In this example, it is defined that the elevenmonth authorized membership token was verified by a user@str3em.com on Jan. 1, 2011. By providing a second token, the first token becomes locked to ownership by the excelsior enabler preventing unauthorized users from reusing the first token without providing the authentication associated with the alternative referenced second token. In the interest of providers of the apparatus delivering this invention, this document will teach a method of a HTTP PUT calculation scheme for automatic royalty billing and administration for the token element used in the invention. Amazon’s DevPay allow developers to attach monetary charges to data services of S3 offered as an embedded component of the apparatus. By using the “PUT” requests parameter, tokens generated by the apparatus are monitored, calculated, and charged to clients of the apparatus provider. For example: the default charge measure for DevPay is \$0.05 for every 1000 PUT requests. By changing the amount to \$100 for every 1000 PUT requests, the apparatus provider is paid a \$0.10 royalty for each token created. Content providers using a connected apparatus like DevPay to deliver and manage digital media distribution do not need to have restrictions on the tokens created as with prior art DRM key providers as DevPay is charged on a pay-as-you-need model on a monthly basis. As a novelty to the apparatus provider, if a content provider fails to pay royalties due, the DevPay hosting will

10

automatically deny token access to all related media products in distribution and restore this verification element when royalties are paid in full.

The authentication element of this invention is at least handled first by the at least one excelsior enabler with a connection to a membership. In the present discussion, the connection is equal to the Internet and the membership is equal to a web service. Further, the web service must be available for two way data exchange to complete the authentication process of this invention. Data exchange with a web service is usually facilitated with a programmable communications console, at most times, will be an Applications Programmable Interface (API). An API is a set of routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services in order to support the building of applications. An API may be language-dependent: that is, available only in a particular programming language, using the particular syntax and elements of the programming language to make the API convenient to use in this particular context. Alternatively an API may be language-independent: that is, written in a way that means it can be called from several programming languages (typically an assembly/C-level interface). This is a desired feature for a service-style API that is not bound to a particular process or system and is available as a remote procedure call. A more detailed description of API that can be used for an apparatus can be found in the book, “Professional Web APIs with PHP: eBay, Google, Paypal, Amazon, FedEx plus Web Feeds”, by Paul Reinheimer, Wrox publishers (2006). A program apparatus, scripts, often calls these APIs or sections of code residing on user computerized devices. For example, a web browser running on a user computer, cell phone, or other device can download a section of JavaScript or other code from a web server, and then use this code to in turn interact with the API of a remote Internet server system as desired. A Graphic User Interface (GUI) can be installed for human interaction or processes can be preprogrammed in a programmable script such as PHP, ASP.Net, Java, Ruby on Rails and others. The authentication element of the invention is usually embedded as a process of the apparatus but could be linked dynamically. In this document, the embedded version using a GUI will be used as reference. The web service equipped with the API is usually a well-known membership themed application in which the users must use an authentic identification. Some example includes Facebook in which as a rule, members are required to use their legal name identities. A reference number or name with the Facebook Platform API represents this information. Other verified web services in which real member names are required such as the LinkedIn API and the PayPal API and even others could be used, but for this discussion, Facebook will be used only as an example of how the authentication element of the invention is utilized. The Facebook API system, as well as others, operates based on an access authentication system called from a connected apparatus (which is usually an Internet powered desktop or browser based application) with an API Key, an Application Secret Key and could also include an Application ID. For example, the Facebook API Application Keys required to establish a data exchange session with the connected apparatus might look like:

---

```
API Key
37a925fc5ee9b4752af981b9a30e9a73gh
Application Secret
f2a2d92ef395cce88eb0261d4b4gsa782
Application ID
51920566446
```

---

## US 8,402,555 B2

11

The collective API keys are usually embedded in the source code of the apparatus, or stored on a remote Internet server, and could be included in the encrypted digital media metadata and inserted on-the-fly into calls made to the API from the connected apparatus. This allows dynamic API connection of the apparatus using keys issued to individual content providers so in the event of a reprimand of a single the individual content provider by the API provider, the collective the individual content providers and the enablers of the connected apparatus are not affected.

Upon an access request of the digital media, the excelsior enabler interacts with the apparatus, usually software or web application, to enter membership credentials in a GUI front-end connected to the API. The membership credentials are usually comprised of a login element comprising a name, phrase, or e-mail address, and a secret password. The credentials can be generated by the enabler or automatically generated by the web service. Once the enabler authenticates their identity with the membership, the apparatus facilitating the data communication can request relevant information to fulfill the process chain of the invention. For example, Facebook API Platform defines members as ID numbers, so if a member's real name is John Doe, then Facebook API ID (also programmatically known as the FBID) would be 39485678. Once the enabler successfully sign in to the GUI element then the apparatus will query the API for at least one electronic identification reference, in this discussion is the FBID. The FBID is received to the permanent or temporary memory of the apparatus to sustain the branding and cross-referencing requirements of the invention. Additional information can be requested according to membership status or connected "friends" of the enabler. Additional information can be made required for successful authentication and includes: a minimum amount of total friends, a minimum amount of female friends, a minimum amount of male friends, a minimum amount of available pictures, a minimum age limit and other custom rules can be defined by the apparatus. An example of how this would work is a content provider can define a minimum of 32 Facebook friends are required to access an encrypted digital media asset offered for sale or promotion. This is achieved by the apparatus handling a access request in which the enabler has not yet acquired access rights by executing and parsing information returned by the Facebook "Friends.get" API command.

XML return example of the Facebook "Friends.get" API command where a plurality of FBID are returned to the apparatus for parsing additional information as may be required to satisfy successful authentication:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_get_response xmlns="http://api.facebook.com/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/
    http://api.facebook.com/1.0/facebook.xsd"list="true">
  <uid>222333</uid>
  <uid>1240079</uid>
</friends_get_response>
```

---

When authenticating a compatible device or machine which may not have access to a connection for the authentication element, a key file or part of the metadata thereof could be made on another connected compatible device or machine and allow the enabler to execute Friends.get API command to collect and store the complete list of a plurality of FBID to the key file or the metadata thereof. The compatible device or machine which may not have access to a connection for the

12

authentication element with an embedded interaction console, usually a user GUI, can request and load the key file or part of the metadata thereof to save the complete list of a plurality of electronic identification references, in this discussion is reference as the FBID, to storage or metadata as part of the compatible device or machine. This step ensures the cross-referencing element requirement of the invention can take place in the event the connection for the authentication element is not present in the compatible device or machine.

Another example is a content provider can allow shared access to friends of the excelsior enabler after a time period, like for example, 90 days. After the 90 day period, when media access is requested using the authentication element by a plurality of secondary enablers, which are usually friends and family of the excelsior enabler, the FBID of the excelsior enabler is cross-referenced with the FBID of the requesting secondary enabler by way of the apparatus ability to execute the Facebook "Friends.areFriends" API command.

XML return example of the Facebook "Friends.areFriends" API command where FBID 2223332 and 2223333 are friends and FBID 1240077 and 1240079 are not friends:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_areFriends_response
  xmlns="http://api.facebook.com/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/
    http://api.facebook.com/1.0/facebook.xsd"list="true">
  <friend_info>
    <uid1>222333</uid1><uid2>222333</uid2>
    <are_friends>1</are_friends>
  </friend_info>
  <friend_info>
    <uid1>1240077</uid1><uid2>1240079</uid2>
    <are_friends>0</are_friends>
  </friend_info>
</friends_areFriends_response>
```

---

Such usability can be important to sustain "fair use" rights of consumers of the digital media to emulate usability found with physical media products such as CD and DVD that can be loaned to friends and family after an inception grace period.

Once the information of the verification and authentication elements is acquired, the apparatus handles the next process of writing the information to the digital media metadata and can include additional information gathered from components of The App. Components of The App can include MAC address from a networking card, CRC checksum of an embedded file or circuit, SOC identifier, embedded serial number, OS version, web browser version, and many other identifiable components as part of The App. For this discussion, the MAC address from a networking card as part of The App will be used as reference of a secondary electronic identification reference. In computer networking, a Media Access Control address (MAC address) is a unique identifier assigned to most network adapters or network interface cards (NICs) by the manufacturer for identification, and used in the Media Access Control protocol sub-layer. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number. It may also be known as an Ethernet Hardware Address (EHA), hardware address, adapter address, or physical address. The novelty of embedding the MAC address along with the FBID of the excelsior enabler is to provide a plurality of electronic identification references in which cross-referencing actions can allow more rapid access to be granted with less interaction from an



US 8,402,555 B2

13

enabler. For example, to retrieve the FBID from Facebook to cross-reference with the FBID stored in the digital media metadata requires the enabler to possibly physically need to enter their login and password credentials to the GUI connected to the apparatus. It may be possible that web browser cookies allow automatic Facebook login by storing an active session key, but the session key is not guaranteed to be active at the time of an access request. While the enabler may not have an issue executing another authentication command, several remote operations could exist to control authentication and access requests separately from each other. The apparatus can execute a programmable retrieval command, usually a GET command, to locate and retrieve the MAC address from an attached or connected networking card. After the FBID is acquired, the MAC address is also acquired to write the plurality of electronic identifications to the metadata of the at least one encrypted digital media asset by; obtaining the decryption key to decrypt the encrypted digital media asset which is usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connected source, usually an Internet server with an encrypted HTTPS protocol. A plurality of MAC addresses can be stored along with the FBID of the excelsior enabler to manage access rights across a plurality of devices. To understand metadata and the uses, metadata is defined simply as to “describe other data”. It provides information about certain item’s content. For example, an image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document’s metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document. Web pages often include metadata in the form of Meta tags. Description and keywords Meta tags are commonly used to describe the Web page’s content. Most search engines use this data when adding pages to their search index. In the invention, the FBID and MAC addresses are written to the digital media asset metadata to prepare for the instant or delayed cross-referencing element of the invention. The same process of writing the information to the digital media metadata is true with secondary enablers allowing the same benefits of cross-referencing.

Cross-referencing, the last element of the invention is used to verify access rights of an enabler of a pre or post personalized encrypted digital media asset. Once an enabler executes an action for access request, the apparatus will obtain the decryption key to first seek the MAC address record. If the MAC address is found, then a cross-reference process is executed by comparing the MAC address retrieved from the metadata of the digital media file with the MAC address retrieved from the networking card connected to the apparatus or The App. If the comparison action proves to be true, then access rights are granted to the enabler. If the comparison fails, then the apparatus can either ask the enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the enabler. In this discussion, the apparatus requires the enabler to participate in communication with the authentication element to provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook API. If the comparison action proves to be true, then access rights is granted to the excelsior enabler and the current MAC address of the networking card as part of The App is appended to the metadata of the encrypted digital media asset and access rights is granted to the excelsior enabler. If the FBID cross-reference fails, then the apparatus can either ask the potential secondary

14

enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the potential secondary enabler. In this discussion, the apparatus requires the potential secondary enabler to participate in communication with the authentication element to provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook “Friends.areFriends” API command to determine if the potential secondary enabler identity is true or false. The determination is in accordance to any possible access grace periods set by the content provider of the encrypted digital media asset. If the comparison action proves to be true, then access rights is granted to the secondary enabler and the current MAC address of the networking card as part of The App and the FBID retrieved are appended to the established metadata information of the encrypted digital media asset and access rights can be granted to a plurality of secondary enablers; unlimited interoperability between devices and “fair use” sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments is achieved.

While the present invention has been described in connection with preferred embodiments, it will be understood by those skilled in the art that variations and modifications of the preferred embodiments described above may be made without departing from the scope of the invention. Other embodiments will be apparent to those skilled in the art from a consideration of the specification or from a practice of the invention disclosed herein. It is intended that the specification and the described examples are considered exemplary only, with the true scope of the invention indicated by the following claims.

What is claimed is:

1. A method for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the method comprising:

receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

authenticating the membership verification token, the authentication being performed in connection with a token database;

establishing connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol, wherein the API is obtained from a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console; and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

2. The method according to claim 1, wherein the membership verification token is one or more of a structured password, a random password, e-mail address, payment system

## US 8,402,555 B2

15

and one or more redeemable instruments of trade for access rights of the encrypted digital media.

3. The method according to claim 1, wherein the branding request being a request from the first user through a data processing device of the plurality of data processing devices, the first user acquiring access rights to the encrypted digital media; or

wherein the branding request being a request from one or more secondary users connected to the first user, the one or more secondary users comprising one or more of human beings or programmed computerized mechanisms in network of the first user; wherein the one or more secondary users are validated by a membership web service.

4. The method according to claim 3, wherein the membership verification token represents verification from content provider to grant access rights to the first user and the one or more secondary users.

5. The method according to claim 1, wherein the encrypted digital media is shared with one or more users according to a membership.

6. The method according to claim 5, wherein the one or more users are a network of recognized human beings using machines or recognized automated computerized mechanisms programmed by human beings, the recognition of the users being validated by the membership status of a membership web service.

7. The method according to claim 1, wherein the encrypted digital media is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from a list of associated tokens stored in the token database for verification.

8. The method according to claim 1, wherein the encrypted digital media is one of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

9. The method according to claim 1, wherein the electronic identification reference is a key file, the key file being uploaded by the at least one communications console for branding the encrypted digital media; thereby giving access to the encrypted digital media.

10. The method of claim 1, wherein the method facilitates access rights authentication for the encrypted digital media, the branding request is an access request, and wherein the read or write request of metadata is performed in connection with a combination of a memory, CPU, server, database, and cloud system;

the access request is generated by either a human user, a machine, or a human programmed computerized device; the access request further comprises a membership verification token and a rights token wherein the rights token is a flag indicating the verification token is successfully verified.

11. The method of claim 2, wherein the membership verification token comprises at least one token selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by one or more of a letter, number, combination of letters and numbers, phrase, authorization, list, interface button or an instrument of trade for access rights of the encrypted digital media.

12. A system for monitoring access to an encrypted digital media, the system facilitating interoperability between a plurality of data processing devices, the system working as a

16

front-end agent for access rights authorization between a plurality of data processing devices, the system comprising:

a first receipt module, the first receipt module receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

an authentication module, the authentication module authenticating the membership verification token, the authentication being performed in connection with a token database;

a connection module, the connection module establishing connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is obtained from a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

a request module, the request module requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user,

a second receipt module, the second receipt module receiving the at least one electronic identification reference from the at least one communications console; and

a branding module, the branding module branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

13. The system according to claim 12, wherein the encrypted digital media is one of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

14. The system according to claim 12, wherein the electronic identification reference is a key certificate file, the key certificate file being uploaded by the at least one communications console for branding the encrypted digital media; thereby giving access to the encrypted digital media.

15. A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:

receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

authenticating the membership verification token, the authentication being performed in connection with a token database;

establishing connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is obtained from a verified web service,

## US 8,402,555 B2

17

the verified web service capable of facilitating a two way data exchange to complete a verification process; requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user; receiving the at least one electronic identification reference from the at least one communications console; and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

16. The computer program product of claim 15, wherein the membership verification token comprises at least one token selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme;

wherein the permission is represented by one or more of a letter, number, combination of letters and numbers, phrase, authorization, list, interface button or an instrument of trade for access rights of the encrypted digital media.

17. The computer program product of claim 15, wherein the computer program product facilitates access rights authentication for the encrypted digital media, the branding request is an access request, and wherein the read or write request of metadata is performed in connection with a combination of a memory, CPU, server, database, and cloud system;

the access request is generated by either a human user, a machine, or a human programmed computerized device; the access request further comprises a membership verification token and a rights token; wherein the rights token is a flag indicating the verification token is successfully verified.

18. The computer program product according to claim 15, wherein the branding request is a request from the first user providing a credential to a membership web service through a data processing device of the plurality of data processing devices, the first user being a human user acquiring access rights to the encrypted digital media.

19. The computer program product according to claim 18, wherein the branding request is a request from one or more secondary users asked to participate in providing a credential to the membership web service connected to the first user, the credential being one generated manually or generated automatically by the membership web service, the plurality of secondary user -comprising one or more of human beings or a programmed computerized mechanism in the network of the first user.

20. The computer program product according to claim 19, wherein the membership verification token represents verification from content provider to grant access rights to the first user and the one or more secondary users.

18

21. The computer program product according to claim 18, wherein the encrypted digital media is shared with one or more secondary users according to a membership status.

22. The computer program product according to claim 21, wherein the one or more secondary users is a programmed and automated machine hosting an operating system that is operated by the first user.

23. The computer program product according to claim 15, wherein the encrypted digital media is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.

24. The system of claim 12, wherein the system facilitates access rights authentication for the encrypted digital media, the branding request is an access request, and wherein the read or write request of metadata is performed in connection with a combination of a memory, CPU, server, database, and cloud system;

the access request is generated by either a human user, a machine, or a human programmed computerized device; the access request further comprises a membership verification token and a rights token; wherein the rights token is a flag indicating the verification token is successfully verified.

25. The system of claim 12, wherein the membership verification token comprises at least one token selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by one or more of a letter, number, combination of letters and numbers, phrase, authorization, list, interface button or an instrument of trade for access rights of the encrypted digital media.

26. The system of claim 12, wherein the encrypted digital media capable of interoperability between a plurality of data processing devices, is further authored by an authoring system comprising:

- a selection module, the selection module selecting one or more media items to form the encrypted digital media;
- a password module, the password module entering a master password which provides access to the encrypted digital media for editing;
- a customization module, the customization module customizing user access panel of the encrypted digital media;
- a database module, the database module connecting the encrypted digital media to a database of membership verification token required for decrypting the encrypted digital media; and
- an encryption module, the encryption module encrypting the one or more media items to create the encrypted digital media.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,402,555 B2  
APPLICATION NO. : 13/397517  
DATED : March 19, 2013  
INVENTOR(S) : William Grecia

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims

In claim 1, column 14, line 49; claim 12, column 16, line 17; and claim 15, column 16, line 63:

the word "connection", in each occurrence, should be changed to --a connection--

In claim 1, column 14, line 52; claim 12, column 16, line 20 and 21; and claim 15, column 16, line 66:

the word "Applications", in each occurrence, should be changed to --Application--

In claim 1, column 14, line 53; claim 12, column 16, line 21; and claim 15, column 16, line 67:

the phrase "obtained from", in each occurrence, should be changed to --related to--

In claim 14, column 16, line 43:

the word "on" should be changed to --one--

In claim 19, column 17, line 47:

the word "-comprising" should read --comprising--

Signed and Sealed this  
Twenty-fourth Day of September, 2013



Teresa Stanek Rea  
*Deputy Director of the United States Patent and Trademark Office*

# EXHIBIT 5



**IN THE UNITED STATES DISTRICT COURT  
FOR THE WESTERN DISTRICT OF TEXAS  
WACO DIVISION**

**WILLIAM GRECIA,**

**Plaintiff,**

**vs.**

**CULLEN/FROST BANKERS, INC.,**

**Defendant.**

**Civil Action No.: 6:21-cv-00016**

**JURY TRIAL DEMANDED**

**COMPLAINT FOR PATENT INFRINGEMENT**

William Grecia brings this patent-infringement action against Cullen/Frost Bankers, Inc. (“Frost Bank”).

**Parties**

1. Plaintiff William Grecia is an individual residing in Downingtown, Pennsylvania.
2. Frost Bank is a Texas corporation, having its principal place of business in San Antonio, Texas.

**Jurisdiction and Venue**

3. This action arises under the patent laws of the United States, 35 U.S.C. §§ 101 *et seq.*
4. This Court has subject matter jurisdiction over this action under 28 U.S.C. §§ 1331 and 1338(a).
5. This Court may exercise personal jurisdiction over Frost Bank. Frost Bank conducts continuous and systematic business in this District; and this patent-infringement case arises directly from Frost Bank’s continuous and systematic activity in this District. In short, this

Court's exercise of jurisdiction over Frost Bank would be consistent with the Texas long-arm statute and traditional notions of fair play and substantial justice.

6. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391(b)(1)-(2) and 1400(b).

**Infringement of U.S. Patent No. 8,402,555**

7. William Grecia is the exclusive owner of United States Patent No. 8,402,555 (the “‘555 patent”), attached hereto as “Exhibit A.”

8. The ‘555 patent is valid and enforceable.

9. Frost Bank infringes claim 16 of the ‘555 patent. Frost Bank uses the Zelle computer product. This product associates Frost Bank customer accounts with a Zelle service account identifier.

10. Attached hereto as “Exhibit B” and incorporated into this complaint as if alleged herein is a claim chart setting forth the language of claim 16 and the accused Zelle computer product that Frost Bank uses.

11. Claim 15 of the ‘555 patent is a computer product: “A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of . . . .” (‘555 patent, col. 16:46-52.)

12. Frost Bank offers its customers—individuals and businesses holding accounts with Frost Bank—a way to make and receive payments digitally. This is the Zelle computer

program product that includes code that facilitates monitoring access to the Frost Bank account holder's money.

13. The claim 16 product has the capability of “receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media . . . .” (‘555 patent, col. 16:53-59.)

14. Frost Bank uses of the Zelle computer program product. The Zelle product is capable of receiving an access request in the form of an email or mobile telephone number through the Frost Bank communications console—Frost Bank’s mobile application.

15. Claim 16’s product has the capacity for “authenticating the membership verification token, the authentication being performed in connection with a token database . . . .” (‘555 patent, col. 16:60-62.) Frost Bank uses the Zelle computer program product, which is capable of authenticating the membership verification token with a Zelle User ID database. In other words, the Zelle product that Frost Bank uses includes a database that authenticates the Frost Bank account holder’s email or mobile phone number.

16. Claim 16: “establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Application Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process . . . .” (‘555 patent, cols. 16:63-17:2.)

17. Frost Bank uses of the Zelle computer program product. This product is capable of establishing an API communication related to the ZELLE RESTful API. The Frost Bank

communications console (i.e., the Frost Bank mobile app) is a combination of a GUI and an API related to the Zelle network.

18. Claim 16 is a product with the capacity of “requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user; receiving the at least one electronic identification reference from the at least one communications console . . . .” (‘555 patent, col. 17:3-8.)

19. Frost Bank uses the Zelle computer program product, which is capable of requesting and receiving a CXCToken (i.e., an account electronic identification reference) from the Zelle network.

20. Claim 16’s product includes the capacity for “branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.”

21. Frost Bank uses the Zelle computer program product, which is capable of writing the Frost Bank account holder’s email or phone number and the Zelle CXCToken to the Zelle Computer Product metadata. By this action, which the Zelle product is capable of performing, Frost Bank enrolls the Frost Bank account holder in the Zelle service.

22. Claim 16, “wherein the membership verification token comprises at least one token selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by one or more of a letter, number, combination of letters and numbers, phrase, authorization, list, interface button or an instrument of trade for access rights of the encrypted digital media.” (‘555 patent, col. 17:12-22.)

23. Frost Bank uses the Zelle computer program product to associate Frost Bank accounts with a Zelle service account identifier. The account holder's email or phone number is a membership permission coupled to a royalty scheme. Once the account holder's email or phone number is enrolled and capable of sending money via the Zelle computer product, the Zelle product counts each instance of sent money as a "transaction." Enrolling the account holder's email or phone number is coupled to a royalty scheme according to each Zelle "transaction."

### **Prayer for Relief**

WHEREFORE, William Grecia prays for the following relief against Frost Bank:

- (a) Judgment that Frost Bank has directly infringed claim 16 of the '555 patent;
- (b) For a reasonable royalty;
- (c) For pre-judgment interest and post-judgment interest at the maximum rate allowed by law;
- (d) For injunctive relief, including a preliminary injunction; and
- (e) For such other and further relief as the Court may deem just and proper.

### **Demand for Jury Trial**

William Grecia demands a trial by jury on all matters and issues triable by jury.

Respectfully Submitted,

Date: January 8, 2021

/s/ H. Artoush Ohanian

H. Artoush Ohanian  
[artoush@ohanianip.com](mailto:artoush@ohanianip.com)  
OHANIANIP  
604 West 13th Street  
Austin, Texas 78701  
(512) 298.2005 (telephone & facsimile)

Matthew M. Wawrzyn (application for *pro hac vice* admission forthcoming)

[matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)

WAWRZYN LLC

200 East Randolph Street, Suite 5100

Chicago, IL 60601

(312) 235-3120 (telephone)

(312) 233-0063 (facsimile)

*Counsel for William Grecia*

# EXHIBIT 6



---

**From:** Maskovich, Jacob  
**Sent:** Friday, February 12, 2021 10:25 AM  
**To:** McClintock, Teri  
**Subject:** FW: Preliminary litigation avoidance discounted patent license offer  
**Attachments:** FNBCT\_Zelle\_555-23.pdf



JACOB MASKOVICH  
Attorney  
BRYAN CAVE LEIGHTON PAISNER LLP - Phoenix, AZ USA  
jamaskovich@bclplaw.com  
T: +1 602 364 7164

Emerging Themes in Financial Regulation 2021 ▶

---

**From:** Chen, George <george.chen@bryancave.com>  
**Sent:** Thursday, February 4, 2021 9:38 AM  
**To:** Maskovich, Jacob <jamaskovich@bryancave.com>; Smith, Cory <cory.smith@bryancave.com>; Kelly, Erin <erin.kelly@bryancave.com>  
**Subject:** FW: Preliminary litigation avoidance discounted patent license offer

Redacted



GEORGE C. CHEN  
Partner  
BRYAN CAVE LEIGHTON PAISNER LLP - Phoenix, AZ USA  
george.chen@bclplaw.com  
T: +1 602 364 7367 M: +1 602 769 7233

Phoenix Patent Law Lawyer of the Year, The Best Lawyers in America 2020; Phoenix Intellectual Property Litigation Lawyer of the Year, The Best Lawyers in America 2019; WTR1000 2020; Top Attorney, Super Lawyers 2020; Power List, Lawyers of Color 2020

---

**From:** Johnson, Warren <Warren.Johnson@earlywarning.com>  
**Sent:** Thursday, February 4, 2021 9:35 AM  
**To:** Chen, George <george.chen@bryancave.com>  
**Cc:** Oppenheim, David <David.Oppenheim@earlywarning.com>  
**Subject:** FW: Preliminary litigation avoidance discounted patent license offer

Redacted

---

**From:** Grecia Family <business@greciafamily.estate>  
**Sent:** Thursday, February 4, 2021 7:49 AM  
**To:** ben.kelly@bakermckenzie.com; john.flaim@bakermckenzie.com

**Cc:** Johnson, Warren <Warren.Johnson@earlywarning.com>; Matt Wawrzyn <[matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)>

**Subject:** [EXTERNAL] Preliminary litigation avoidance discounted patent license offer

[RECEIVED FROM AN EXTERNAL SENDER]

---

Dear Mr. Kelly, Mr. Flaim:

Your client First National Bank of Central Texas has a pending Zelle action.

Your client qualifies for a "small entity" pre-litigation blanket license discount that is an alternative to the -price per user- license negotiated after week 8 of the WDTX Case Management Conference.

Please contact Mr. Grecia's lead attorney to discuss the opportunity to establish the discounted blanket license offer of \$2,750,000.

Please contact Mr. Grecia's lead attorney Mr. Matt Wawrzyn ([matt@wawrzynlaw.com](mailto:matt@wawrzynlaw.com)) within 15 business days of this correspondence (by February 15, 2021) to begin the license process.

Please be advised that the -price per user- calculation divided upon a hypothetical negotiation model of \$255 per year, per user will resume after February 15, 2021. Please see the following licensing calculus reference here: <https://www.earlywarning.com/press-release/zeller-closes-2020-record-307-billion-sent-12-billion-transactions>

CC: Early Warning Services LLC as the possible indemnifying party.

Thank You,

*Grecia Estate & Trust*

Grecia Family Estate & Trust

**P:** (212) 390-0355

**E:** [business@greciafamily.estate](mailto:business@greciafamily.estate)

CONFIDENTIALITY NOTICE: The contents of this email message and any attachments are intended solely for the addressee(s) and may contain confidential and/or privileged information and may be legally protected from disclosure. If you are not the intended recipient of this message or their agent, or if this message has been addressed to you in error, please immediately alert the sender by reply email and then delete this message and any attachments. If you are not the intended recipient, you are hereby notified that any use, dissemination, copying, or storage of this message or its attachments is strictly prohibited.

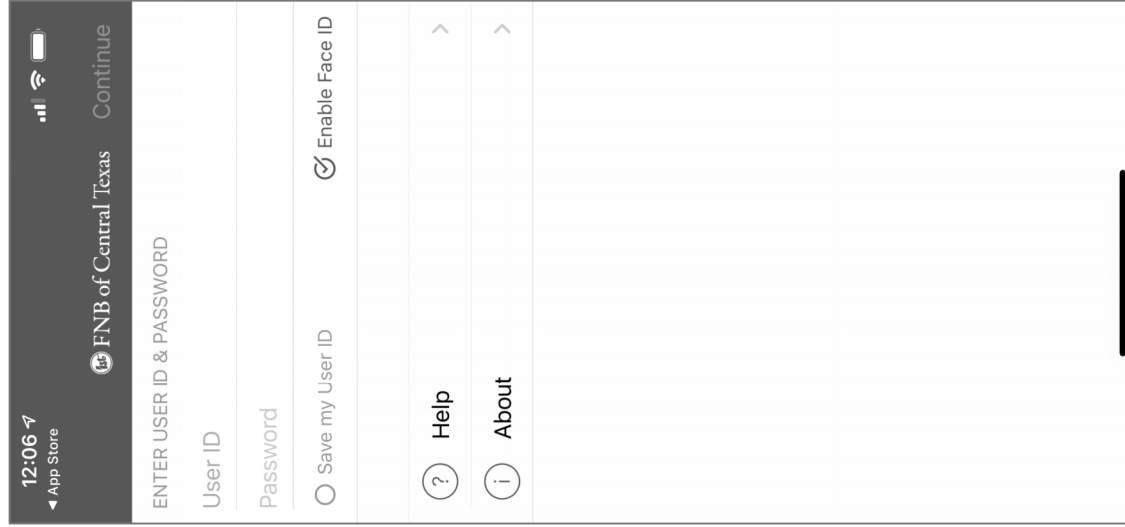
U.S. Patent #8,402,555: FNBCT With Zelle

Computer Readable Medium Claim (Mobile App)

Claim 23	Analysis	Select Evidence
<p>The computer program product according to claim 15, wherein the encrypted digital media [Sullivan claim construction order meaning: computer facilitated data] is associated with an identifier stored in a database, the identifier being cross-referenced with a corresponding token from the list of associated tokens stored in the token database for verification.</p> <p>A computer program product for use with a computer, the computer program product comprising a non-transitory computer usable medium having a computer readable program code stored therein for monitoring access to an encrypted digital media, the method facilitating interoperability between a plurality of data processing devices, the computer program product performing the steps of:</p>	<p>The FNBCT mobile application is a computer program product for use with a (mobile) computer to associate FNBCT accounts with a Zelle service account identifier for cross-reference by writing and cross-referencing a Zelle web service CXCToken (see page 6) to the FNBCT computer product metadata.</p>	<div><div><b>The First National Bank</b> of Central Texas</div><div><div><b>Zelle®</b></div></div><div><b>zelle</b></div><div>THIS IS HOW MONEY MOVES®</div><div>Safely send and receive money with <i>Zelle</i> through The First Mobile banking app.</div><div><div><b>The First Mobile</b> The First National Bank of Central Texas Designed for iPhone ***** 3.4+11 Ratings Free</div><div></div></div><div>Source: Apple App Store</div><div><b>FNBCT - The First Mobile</b> First National Bank of Central Texas Finance ***** 4.6 Contains Ads Everyone This app is compatible with some of your devices You can share this with your family. <a href="#">Learn more about Family Library</a> <a href="#">Add to Wishlist</a> <a href="#">Install</a></div><div>Source: Google Play Store</div></div>

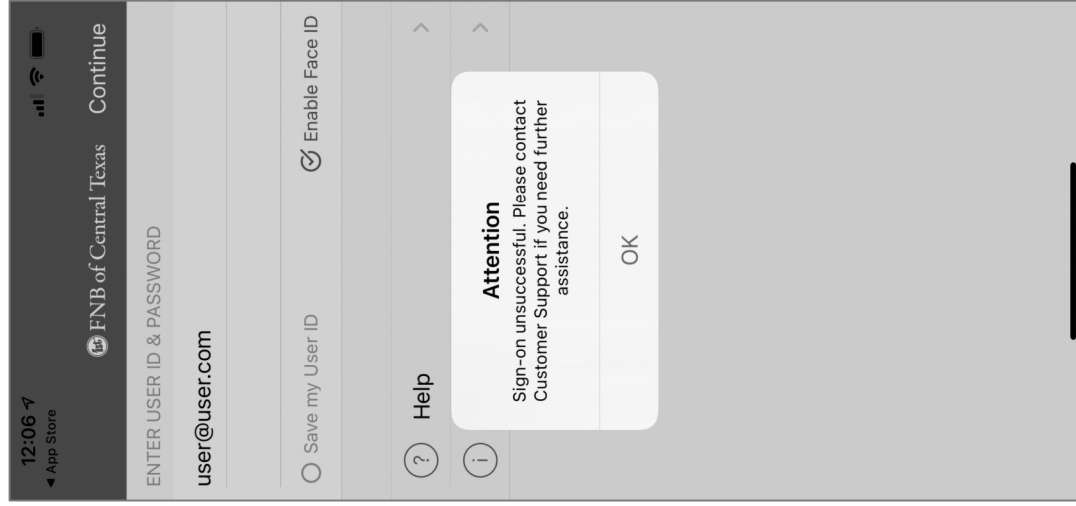
receiving an encrypted digital media access branding request from at least one communications console of the plurality of data processing devices, the branding request being a read or write request of metadata of the encrypted digital media, the request comprising a membership verification token provided by a first user, corresponding to the encrypted digital media;

FNBCT computer product receives an access request by receiving a membership verification token through the FNBCT computer product's communication console.



FNBCT computer product authenticate the membership verification token with a User ID database.

authenticating the membership verification token, the authentication being performed in connection with a token database;



"We expose our API system to the banks when they join our network. At this point we are focused on delivering the service to the banks so that we can have the most robust and secure network available. In the future we may look at opening our APIs more broadly, but at this point we are exposing them to the banks who are the part of the network," said Milica.

#### About Early Warning

Early Warning provides risk management solutions to a diverse network of 2,300 financial institutions, government entities and payment companies, enabling businesses and consumers to transact securely and conveniently. Owned and governed by Bank of America, BB&T, Capital One, JPMorgan Chase and Wells Fargo, Early Warning's unique business model facilitates a data exchange system based on collaborative, shared intelligence. For 25 years, the company has worked with organizations of all sizes to advance collaborative risk management and fraud prevention. For more information please visit .

Source: <https://lelstalkpayments.com/interview-with-early-warning-and-clearxchange-the-most-powerful-bank-focused-alliance-in-the-authentication-and-digital-payments-industry/>

## ZELLE RESTful API DOCUMENTATION

### organization ids.

#### participantID

A 64-byte string used to uniquely identify a Zelle participant.

**NOTE:** This value corresponds to the ID of the Zelle participant in the Zelle Shared Directory. This value may or may not be the same as the partner ID of the Zelle participant. When possible, it is recommended that the participant ID match the partner ID.

#### partnerID

A 20-byte string used to uniquely identify a Zelle participant in FTM.

**NOTE:** This value corresponds to the ID of the Zelle participant in the FTM Shared Directory. This value may or may not be the same as the participant ID of the Zelle participant. When possible, it is recommended that the partner ID match the participant ID.

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

Wayback: [https://web.archive.org/web/20180214204352/http://www-](https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1)

[01.ibm.com/support/docview.wss?uid=swg27050366&aid=1](http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1)

FNBC T computer product establish an API communication related to the ZELLE RESTful API.

establishing a connection with the at least one communications console wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) protocol wherein the API is related to a verified web service, the verified web service capable of facilitating a two way data exchange to complete a verification process;

FNBCT computer product request and receive a CXCToken (e.g., an account electronic identification reference) from the ZELLE RESTful API communication.

requesting at least one electronic identification reference from the at least one communications console wherein the electronic identification reference comprises a verified web service account identifier of the first user;

receiving the at least one electronic identification reference from the at least one communications console;

#### WEB SERVICES

##### CREATE (POST)

This service returns the location (URL) of the newly-created CXCToken with an HTTP status of 201 (CREATED) if executed successfully, or one of the HTTP 4xx error codes if an error occurs.

<https://<servername:serverport>/fxh/svc/cxctokens/>

##### DESCRIPTION

The CXCToken.Create web service registers a token for a participant present in the FTM database.

To register a new participant token, the FTM web service checks to ensure that the maximum number of active tokens has not been reached and, if not, invokes four web services. First, the web service invokes the Zelle Match Recipient to determine if the token already exists. If the token does not already exist, the CXCToken.Create web service calls the Client Service to refresh the fraud detection data provided from the user interface. Next, a call is made to the fraud application to perform fraud detection, storing the fraud check result in the FTM database. If no fraud is detected, the web service creates a payment profile ID and invokes the Zelle Change Token Status web service. For a new participant token, Zelle creates the payment profile and returns a response to FTM to indicate that the participant token was successfully added. Then, the CXCToken.Create web service updates the payment profile and associated token to the FTM database and returns a response to the user interface to indicate that the participant token was successfully added. FTM then calls the Client Service to notify the participant of successful token registration.

24

© Copyright IBM Corp. 2017, 2018.

^Page 24:

Source: <http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>  
Wayback: <https://web.archive.org/web/20180214204352/http://www-01.ibm.com/support/docview.wss?uid=swg27050366&aid=1>

#### CXCTOKEN

A CXCToken entity represents a Zelle token and its associated Zelle payment profile. The base URL for this service is

<https://<servername:serverport>/fxh/svc/cxctokens/>

#### ATTRIBUTES

##### KEY

partnerID : partnerType : participantID : token : paymentProfileID

^Page 21

#### token

A 255-byte string that is a normalized value containing either the email address or mobile phone number of a Zelle participant.

^Page 7, definition of "Token"



and branding metadata of the encrypted digital media by writing the membership verification token and the electronic identification reference into the metadata.

FNBCT computer product write (e.g., enroll user to Zelle) to associate the verification token (email or mobile number) and the Zelle CXCToken for cross-reference with the computer product user account data that associates with users Zelle account data.

## **I sent money to someone & they never received it. What now?**

First, check the payment status within your payment activity in your bank's online or mobile service, or within the Zelle app. If the payment status is pending, the recipient may not have enrolled their mobile number or email address to receive the payment. If the payment status is completed, then the money is already available in the recipient's bank account, or will be within 3 business days if the recipient recently enrolled. If you aren't sure of the status of your payment, contact customer support at 254 772-9330.

Source: <https://www.fnbct.com/getzelle/>

# EXHIBIT 7



US 20070156726A1

(19) **United States**(12) **Patent Application Publication****Levy**(10) **Pub. No.: US 2007/0156726 A1**(43) **Pub. Date:****Jul. 5, 2007**(54) **CONTENT METADATA DIRECTORY SERVICES**(52) **U.S. Cl.** ..... **707/100; 709/238**(76) Inventor: **Kenneth L. Levy**, Stevenson, WA (US)(57) **ABSTRACT**

Correspondence Address:  
**DIGIMARC CORPORATION**  
**9405 SW GEMINI DRIVE**  
**BEAVERTON, OR 97008 (US)**

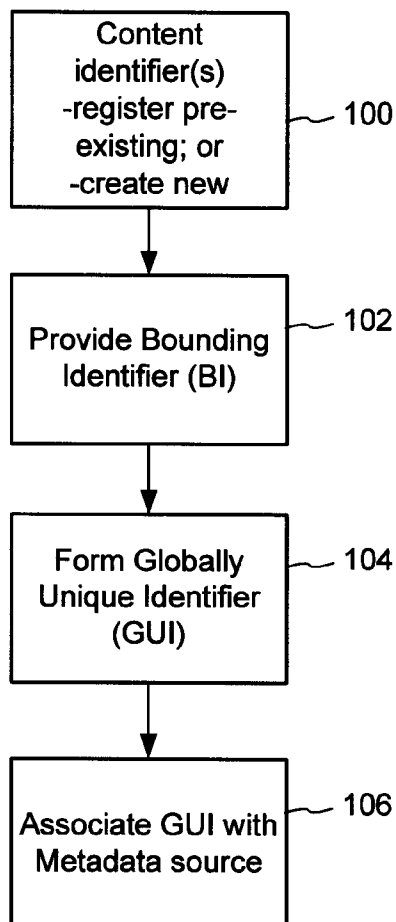
The content metadata directory system connects consumers of identified content to managed metadata databases and other digital resources. The system manages links between identifiers in content objects and metadata sources. It supports a variety of different type of content identifiers and allows for overlap among different content identification schemes. One method of associating a content object with metadata uses a combination of a content identifier and a bounding identifier to enable handling of disparate sets of content identifiers for content objects with potentially conflicting content identifiers. The method receives a content identifier for a content object from among a set of content identifiers and provides a unique bounding identifier for the set of content identifiers. This unique bounding identifier is used in combination with the content identifier to form a globally unique identifier for the content object. This globally unique identifier is associated with a metadata source, which enables routing of a user to the metadata source.

(21) Appl. No.: **11/614,921**(22) Filed: **Dec. 21, 2006****Related U.S. Application Data**

(60) Provisional application No. 60/753,257, filed on Dec. 21, 2005. Provisional application No. 60/747,408, filed on May 16, 2006.

**Publication Classification**

(51) **Int. Cl.**  
**G06F 7/00** (2006.01)  
**G06F 15/173** (2006.01)



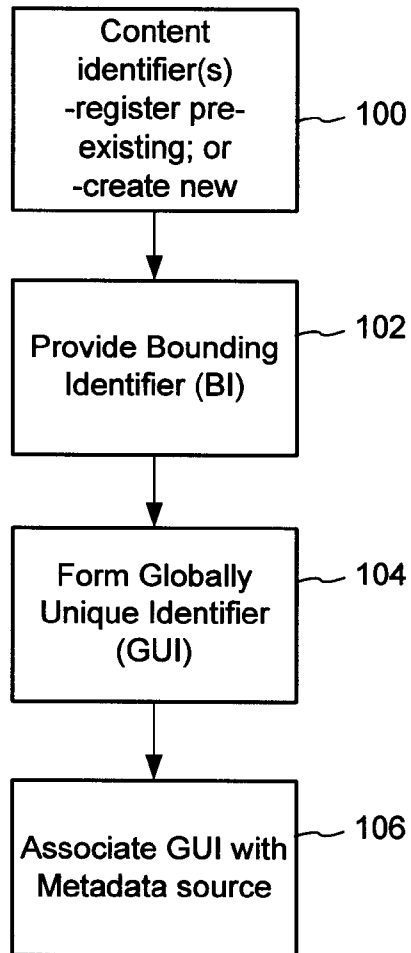


FIG. 1

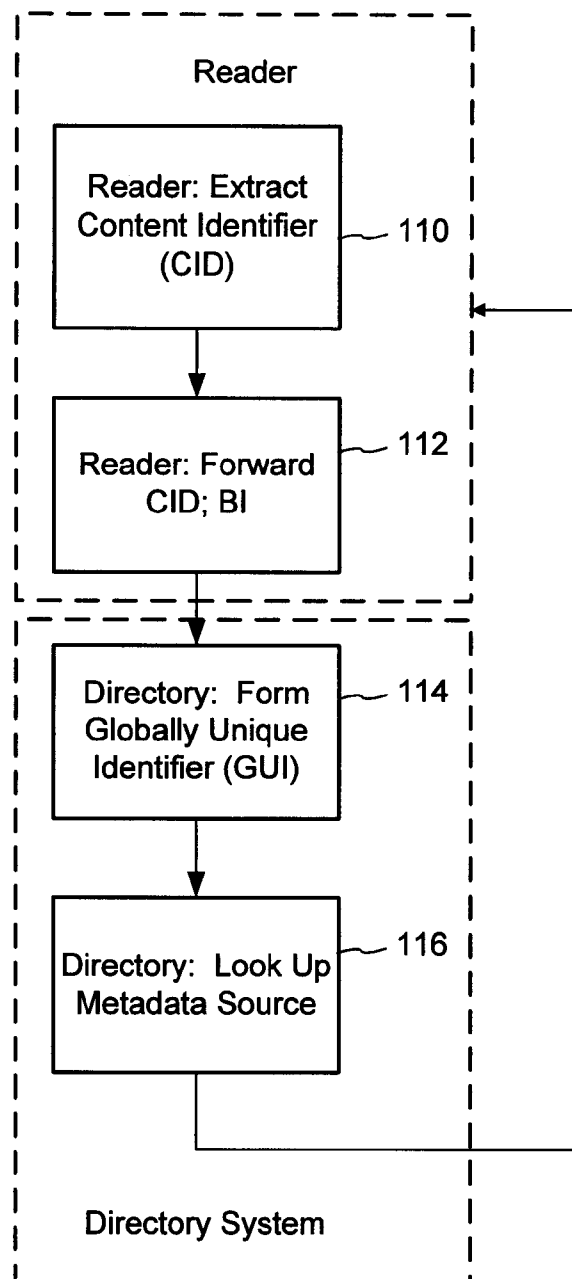


FIG. 2

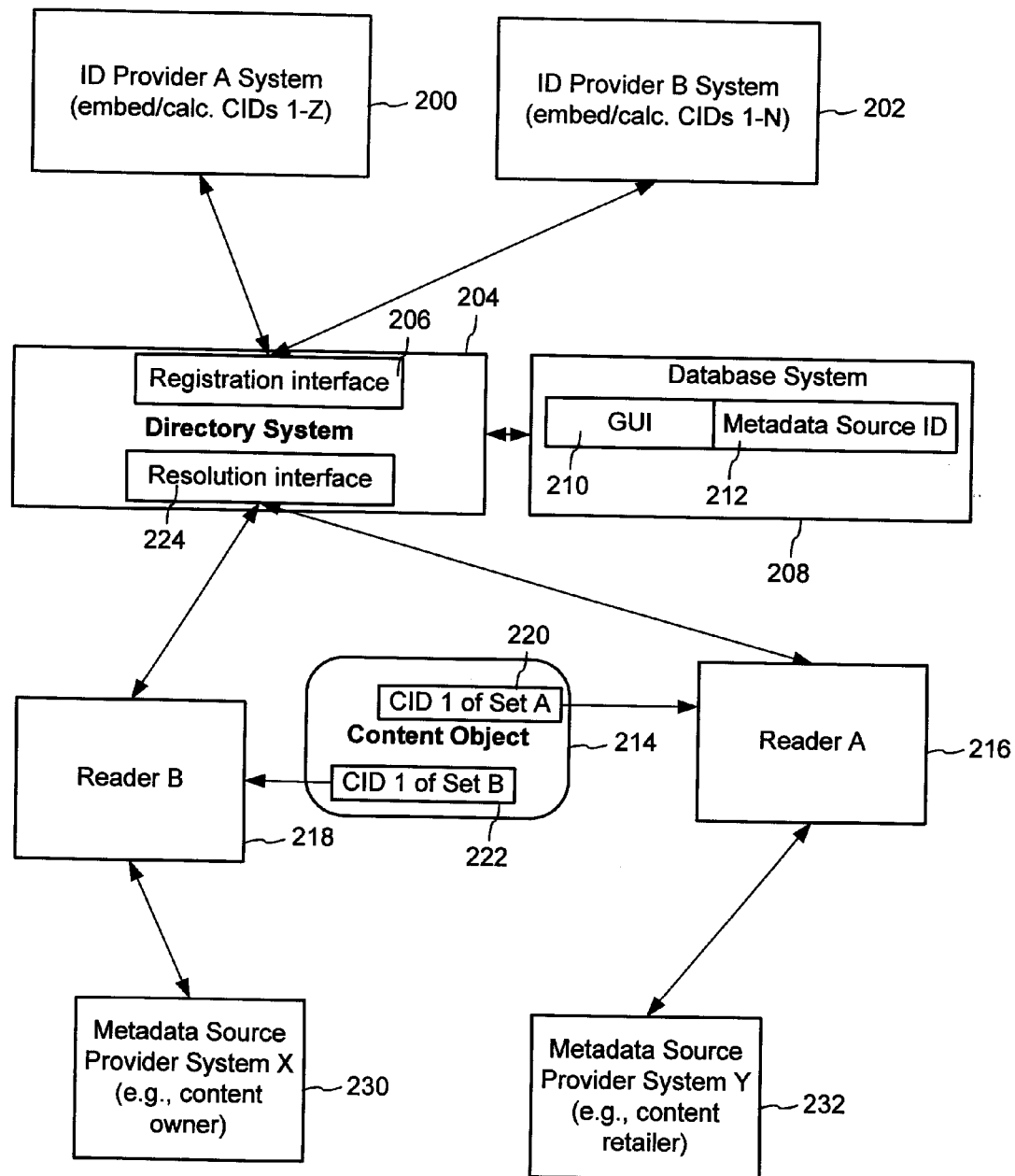


FIG. 3

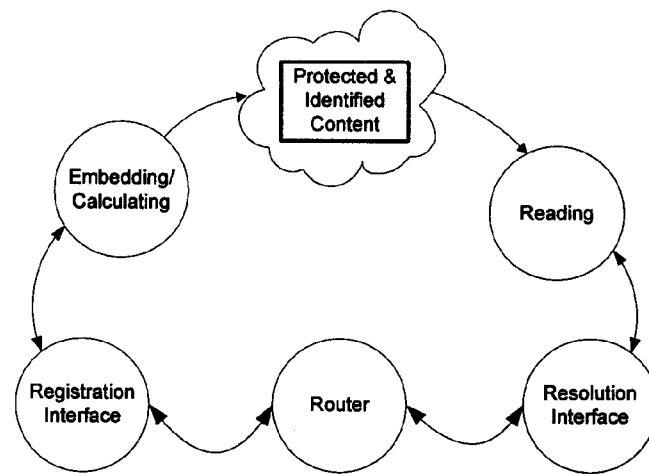


Fig. 4

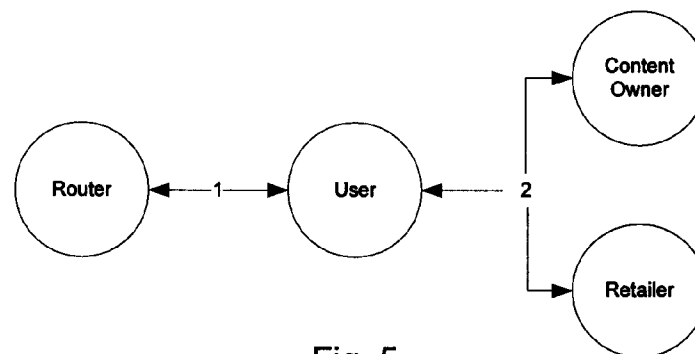


Fig. 5

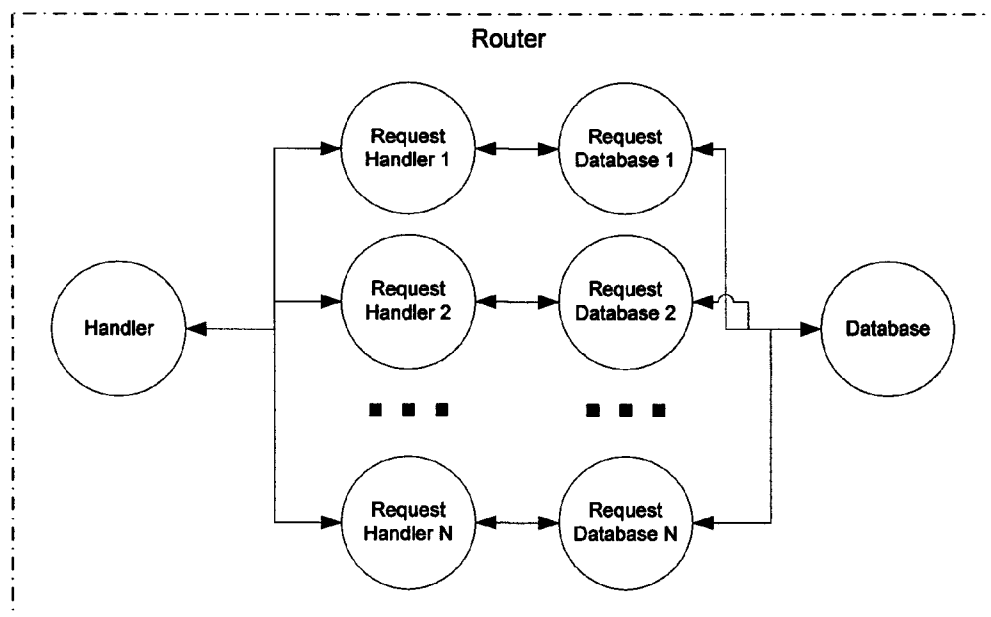


Fig. 6

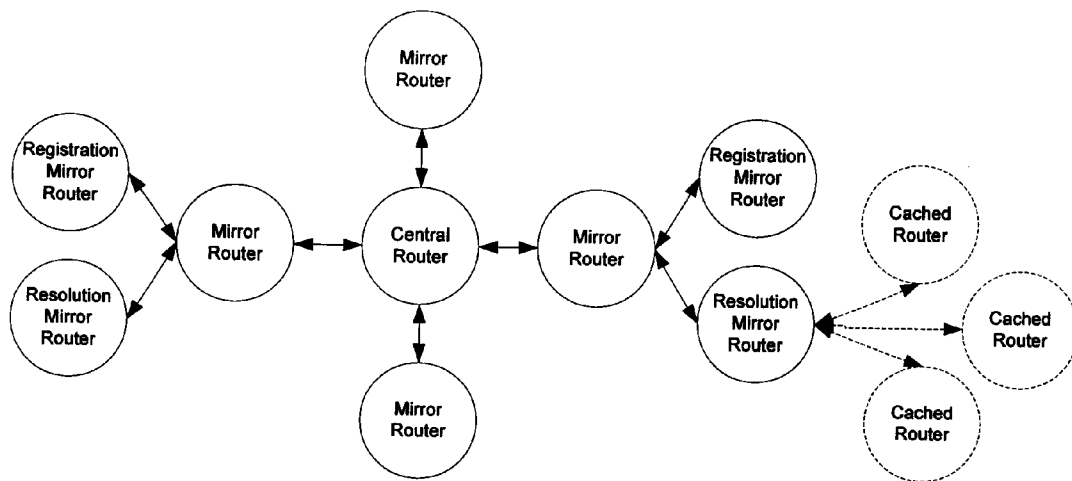


Fig. 7

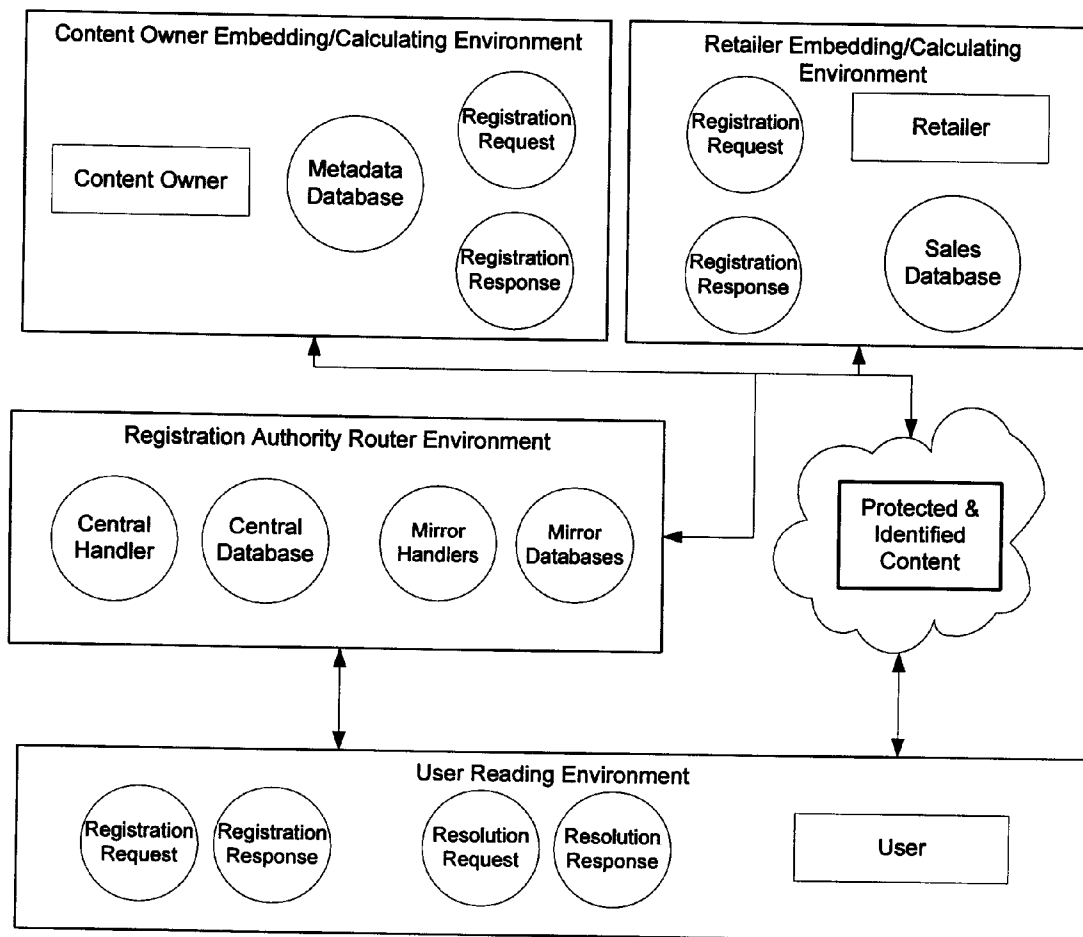


Fig. 8



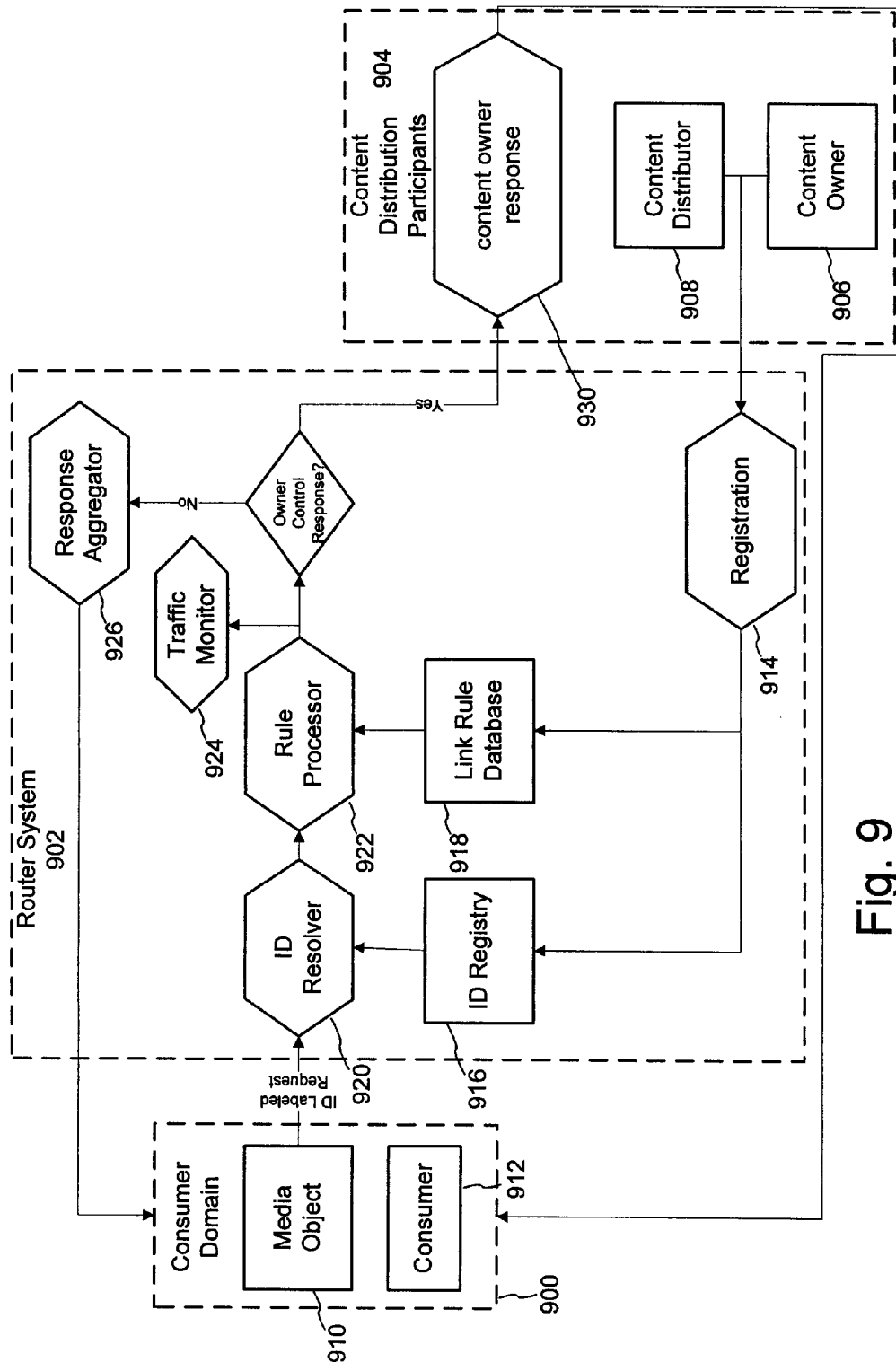


Fig. 9

US 2007/0156726 A1

Jul. 5, 2007

1

**CONTENT METADATA DIRECTORY SERVICES****RELATED APPLICATION DATA**

[0001] This application claims the benefit of U.S. Provisional Applications 60/747,408, filed May 16, 2006, and 60/753,257, filed Dec. 21, 2005, which are hereby incorporated by reference.

**TECHNICAL FIELD**

[0002] The invention relates to methods and systems for associating content, including both physical and electronic objects, with metadata through networks.

**BACKGROUND AND SUMMARY**

[0003] This document uses the terms 'content,' 'media' and 'media content' interchangeably to refer to pictures, music, movies and all of their sundry creative brethren which collectively might fall under the grand umbrella named 'creative works'. In general, a given creative work can be thought of as an entity or 'content object', and either the creative work stands alone with the sole companion of a name or an identification number, or it is somehow duplicated and packaged for a very wide variety of distribution methods and channels.

[0004] In primarily the latter case, the concept of 'metadata' has gained popularity, referring to additional creative works which have some form of explicit or implicit relationship to a given singular content object. Metadata generally refers to information associated with a content object. Typically, metadata is often associated with multimedia content, like images, and video and audio programs, and is used to refer to information about the content, such as its source, owner, content title, etc. One of the simpler forms of metadata might be bit-fields describing additional information about a content object such as an author's name or a category that the content object might naturally fall under. More complicated forms of metadata might be 'pointers' or addresses (e.g. URLs) of related content objects which, by reference, enable a user or consumer to easily access that second content object. Most generally, a key concept from a content provider and/or content distributor's point of view is that metadata can form an instant electronic relationship between a consumer experiencing one of their distributed content objects and themselves.

[0005] In this document, metadata refers to a broad class of information relating to a content object, and it applies to a broad class of content objects, including both physical and electronic objects. Metadata also includes an instruction or set of instructions (possibly distributed over one or more devices) that is executed by machine or machines to perform a behavior associated with an object (e.g., perform an on-line transaction, transmit or transfer content, authenticate/verify a user, content, access token, update/patch a program etc.). Metadata can be formatted and stored in a variety of formats. One format is XML, but there are others. The metadata for a particular content object may be distributed over different storage devices. In such a distributed storage approach, the metadata in one location includes references to metadata in other locations (such an index, pointer, address, URL, etc.).

[0006] One aspect of the invention relates to the technical infrastructure, including metadata routing and associated

network services that ensure this electronic relationship can happen in the first place, and that this relationship can lead toward secondary revenue generation opportunities that may some day rival and eclipse primary media distribution revenue generation.

[0007] The business of selling packaged media or otherwise delivering specific media to a targeted audience has a long history of monetizing the primary delivery of that media. Selling records or selling 30 seconds of advertising on a television show or selling tickets to a movie each fit into the primary distribution monetization business model. The growth in the Internet and the flowering of various digital distribution channels certainly has complicated the description of primary media distribution itself, but the general notion of "packaging up" a creative work and delivering it for some explicit compensation strategy remains intact.

[0008] In a kind of direct extrapolation of the primary distribution model, Digital Rights Management (DRM) inventions and approaches have at least a decade worth of effort, design, trial, partial successes and valuable lessons now under their collective belt. Not least in these lessons are the behind-the-scenes business community wrestling and clashes of Titans surrounding the question on what might be considered a core property of DRM approaches: "who owns the standard . . . who owns the channel . . . who ultimately owns the consumer relationship?" Phrased in this way, there can be little mystery why uniform global standards are possibly decades away still.

[0009] Two somewhat different forces have arisen in the past decade or two which have not been entirely harmful to classic primary media distribution monetization, but they have nevertheless put significant pressure on businesses which rely on primary distribution revenue to look for secondary methods of monetization. At the very least, these forces have led toward fundamental changes of strategy in how primary distribution methods are exploited.

[0010] The first force is the ease with which creative works can be copied and re-distributed in an unauthorized fashion. The second force is the advent of highly distributed media distribution channels and the equally highly distributed end-devices used to experience a creative work, most certainly including mobile devices. Though these two forces are rather different from each other and though each has been partially transformed into "opportunity" by ever-entrepreneurial efforts and companies, the fact remains that both forces are unstoppably disrupting traditional approaches to the monetization of primary media distribution.

[0011] The relative ease with which creative works can be copied has been a primary fuel in creating the now familiar notion of peer-to-peer networks where folks not only share pictures from family vacations but also the latest movie they enjoyed last night. Untold years of technological ponderings and industry standards initiatives have sought to re-establish the core role of primary packaged media distribution and its associated monetization, also not without some success, but the genie does seem to be rather out of the bottle for those seeking to re-create the good old days of creation-to-consumption monetization. Coordinated primary and second monetization strategies and cash-generating mechanisms are inevitably here to stay, most likely co-opting the second force of highly distributed distribution channels and consumption devices. Of particular note is the up and coming

US 2007/0156726 A1

Jul. 5, 2007

2

'mobile' media consumption trend where ubiquitous connectivity meets ubiquitous delivery.

[0012] Still missing in this inevitable balancing of primary and secondary monetization methods are the critical details of the secondary monetization methods and systems, as well as the impact of their existence on primary distribution methods and strategies. In other words, how can secondary monetization work (beyond peddling primary distribution of ring tones), and how can primary distribution production processes be seamlessly modified to put the overall media industry profit and revenue lines back onto positive and strongly growing paths? This document describes a routing system and method, along with detailed hardware and software descriptions of the system and network components which can make it work in a tremendously complicated media distribution and consumption universe.

#### Managing the Fountain of Metadata

[0013] Once a distributor has accomplished primary delivery of a media object to a consumer, the next best thing to "manage" is the metadata services that add value to that delivered object. This is not the slippery slope named "control" which is a word written on many a tombstone of the last decade's worth of packaging and monolithic DRM-system approaches to media consumption behaviors, it is fundamentally about managing the highest quality relationship to the individuals or groups who are naturally attracted to the media content in the first place. The initial packaging up of "good stuff" metadata into the primary delivery of the original media content was one of the main reasons a consumer will pay a modest amount for the officially sanctioned media, but it will be the ongoing access to high quality metadata, group affinities and seamless access to related media which will compel honest consumers to be honest . . . they will simply get more value for their time and money that way.

[0014] Clearly, combining a media object with static metadata or even "static links" to inherently dynamic web-based content is a developed art at this point and somewhat accounts for the ongoing vitality of primary distribution channels over some P2P network channels. In other words, packing in "good extra stuff" still sells records and movies and pictures. P2P copies can try to keep up, but the rightful distributor has a leg up on the availability of legitimately compelling content, precisely because they are the legitimate owners or distribution rights-holder.

[0015] In some advertising-based business models, content objects are distributed for free or reduced cost and provide a vehicle for advertising revenue. These models provide an opportunity for content owners to monetize content by conveying advertising within the content. Yet, to capitalize on this opportunity in distribution networks like wireless networks and the Internet, there is a need for mechanisms to tie the content consumption to revenue opportunities, such as linking the content to electronic transactions to buy related content or products and services advertised within the content.

[0016] Fortunately, there are a few common denominators of all media distribution and media consumption that will never go away and which get to the heart of ensuring a stable relationship between content providers and content consumers. One common denominator is simply identity of the

content itself. Another common denominator is the existence of basic business rules and legal frameworks which collectively define the common sense notion of "legitimate distribution and consumption" and its associated notion of return on investment to the content providers. A third common denominator is the near-universal desire of consumers to have access to the best information related to the content being consumed. And finally, there is a fourth common denominator that a content provider wants to own the "rules of relationship" associated with the content they distribute: Business rules and contracts should define how legitimate and consumer-friendly metadata relationships are carried out.

[0017] One aspect of this invention refers to this notion as the title of this section indicates: managing the fountain of metadata that a consumer wants and will eventually expect. The other side of this coin is classic business principle behind satisfying customers: delivering the highest quality "rewards" in managing this fountain will ensure repeat business.

[0018] The raw mechanics of how this management can happen in the global cacophony of media flow provides the technical foundation for these emerging content distribution and monetization models. The careful reader will see that the described system and network mechanisms are fully complementary to DRM-based approaches in the "digitally contained" world of the Internet and classic dedicated media channel delivery networks on the one hand, and fully able to deal with complexities and steep growth of mobile device consumption on the other hand.

[0019] In one embodiment illustrated in FIG. 9, a routing system includes two primary processing engines that are used to link a media object held by a consumer to a source of metadata. The first engine, including the ID resolver and registry components, can be described as the lingua franca of identification systems, methods, technologies, etc. It can also be effectively described as a virtual "DNS for Content Objects," as this identification engine respects any and all native or monolithic approaches to content identification. Hence the qualifier "virtual" in front of DNS. Its function is to resolve a content ID based on identifying information originating from disparate content identification systems.

[0020] The second engine, the rules database and processor, determines where to re-direct the consumer based on the resolved content ID. This rules engine facilitates secondary revenue generation opportunities because it further enables the system to tailor the metadata response to provide related content, products and services. The content provider universe is complex and often has a wide variety of business interests at play. Such interests are most often encapsulated in contracts between various entities, including the artists which create works in the first place, and such contracts can be extended to the detailed rules of metadata response to normal and/or pro-active metadata requests during media consumption sessions. The quality of the response to the consumer, and the opportunity to direct consumer's to additional revenue generating activities including classic eyeballs/advertising pathways, can be enabled by this rules engine.

[0021] FIG. 9 shows the first and second engines as being part of a router system, these engines can be partitioned and distributed over devices and controlled by different partici-

US 2007/0156726 A1

Jul. 5, 2007

3

pants. The rules processor and database may be partitioned from the router system and implemented in separate instantiations, each controlled by a different participant. In this case, for example, the resolver re-directs a consumer to a rules engine under the control of a participant linked to the object via the ID registry, and this rules engine, in turn, executes a rule that determines the metadata response for the consumer (e.g., a URL or set of URLs to particular metadata). The rules processor may also be executed, at least in part on a device under the client's control. In this case, for example, a set of URLs linked to the content object via the ID registry are returned to the client, which in turn, executes rules to determine the metadata response tailored to the consumer.

[0022] The good old days of selling packaged media is still with us. The content being sold is now the seed for an ongoing relationship in ways that classic "branding" could barely fathom. This disclosure details how these two core engines can be built and operated for the good of content providers and content consumer's alike.

#### Managing the Relationship Between Metadata and Content Objects

[0023] As noted, metadata plays an important role in managing and facilitating transactions in content objects. Some significant examples include the use of metadata in digital distribution of content, electronic commerce, and on-line searching and organization of vast stores of data (e.g., the Internet). As the digital world proliferates and there are numerous transactions in content objects, there is a compelling need to manage the association of metadata and content objects.

[0024] This need is not confined to the digital realm. Because humans live in the physical and analog realm, there will always be a need for efficient schemes for crossing back and forth between the digital and analog realms. In particular, physical objects have corresponding metadata just as electronic objects do. For example, products have corresponding metadata in the form of product information, manuals, catalogs of related products, etc. Printed objects have metadata in the form of electronic versions of the object, ownership, source, time and location of creation, etc. Physical objects link to their metadata via an identifier on or derived from the product or related documentation (e.g., packaging, labels, etc.). Metadata management technologies, thus, need to be able to support this physical/electronic interface. Emerging applications include linking physical objects to Internet related information and electronic transactions as described in U.S. Pat. Nos. 6,947,571 and 6,505,160 and International Patent Application WO 97/43736, which are incorporated by reference.

[0025] A significant aspect of managing metadata of disparate content objects is providing effective technologies and schemes for content identification. This is important in the digital realm, where there are many potentially conflicting content identification technologies and architectures. It is also important for managing metadata for physical objects in the digital realm, where identifiers extracted or derived from physical objects provide a form of digital identity of the physical object in the digital world. The metadata systems and methods in this document are designed to work with identification systems that operate in the digital realm only, as well as ones that span the digital and physical

realms. The latter category includes identification methods that derive content identifiers from an electromagnetic signal captured from an analog representation of audio or images (e.g., a digital watermark, content fingerprint, visual symbology, pattern recognition, voice recognition, OCR, etc.) as well content identifiers read via electromagnetic readers of physical data carrier devices like magnetic stripes (and other magnetic data carriers), RF ID tags, smart cards, etc. For example, physical object can be identified via RFID tags, as described at [www.epcglobalinc.org](http://www.epcglobalinc.org) and in the overview document ([www.epcglobalinc.org/news/EPCglobal\\_Network\\_Overview\\_10072004.pdf](http://www.epcglobalinc.org/news/EPCglobal_Network_Overview_10072004.pdf)), which is incorporated by reference.

[0026] Such content identification technology provides a means to identify content objects, but the variety of content identification schemes and formats poses compatibility and interoperability challenges. Moreover, such systems cannot provide useful information without an effective system and method to associate various identifiers with the appropriate metadata.

[0027] The problems are multifold and created by the fact that digital distribution separates content from packaging, new 1-1 marketing opportunities are minimally being utilized, and digital distribution is moving forward with proprietary channels that make the value chain more complex rather than simpler.

[0028] For instance, once content is digitized information typically carried on physical packaging is lost from the content. Digital downloads are partial products, "files without packaging and related metadata". Metadata loss is central to issues surrounding digital content management, piracy and e-commerce. Manual population of multiple distribution channels' metadata repositories gives rise to human error and inaccurate metadata.

[0029] Marketing opportunities are being lost once content is distributed since content owners and retailers lose contact with the consumer. Loss of 1-1 marketing capabilities, especially with digital distribution gaining traction, leads to loss of potential revenue.

[0030] Channels of distribution (e.g., online music retailers, podcasts, social networking sites, user-generated content sites, and P2P networks) and the number of digital derivatives (ring tones, mobile videos, etc.) stemming from a single digital product are increasing. Accurate and effective content identification is an absolute requirement to manage content effectively. Content owners are currently evaluating their metadata repositories trying to understand how to streamline in a manner that is cost-effective.

[0031] Proprietary content identification and metadata systems complicate, rather than simplify, the value chain. Content is embedded with many identifiers that do not interoperate. A few proprietary systems are linking content to metadata without input of the content owners, thus increasing the number of value chain participants.

[0032] Previous initiatives to create a central content metadata repository have failed due to proprietary, political and technical issues of creating a repository rather than directory service. Content owners and retailers want to manage their proprietary metadata and participate in building the relationship with the consumer. Third party metadata companies, and related companies, such as those that orga-



US 2007/0156726 A1

Jul. 5, 2007

4

nize, classify, search and provide search results based on metadata (such as search engine providers), stand to profit from potential unauthorized use of content owners' metadata.

[0033] This document describes systems and methods for associating metadata with content objects. It describes embodiments of novel routing methods and systems referred to as content metadata directory services.

#### Globally Unique Identifier Scheme

[0034] One novel method of associating a content object with metadata uses a combination of a content identifier and a bounding identifier to enable handling of disparate sets of content identifiers for content objects with potentially conflicting content identifiers. The method receives a content identifier for a content object from among a set of content identifiers. It provides a unique bounding identifier for the set of content identifiers. This unique bounding identifier is used in combination with the content identifier to form a globally unique identifier for the content object. This globally unique identifier is associated with a metadata source, which enables routing of a user to the metadata source.

[0035] This approach effectively manages cases where an ID provider pre-assigns a set of content identifiers to objects, and then later registers them in our novel directory system. It also manages cases where the directory system assigns the content identifier prior to insertion of the content identifier in the content object by an ID provider.

[0036] As set forth in the CMDS embodiments, the unique bounding identifier may comprise an ID provider identifier. For example, RFID, EPC, digital watermarking and fingerprinting technology providers can serve as ID providers in the system with overlapping content ID numbers, but unique ID provider IDs. Each ID provider may also use an ID version to distinguish different versions of its technology or content ID spaces.

[0037] After appropriate registration, the directory system is used to route users to a metadata source. For example, the user (e.g., the reader executing on the user's device) provides the content ID from the content object and the bounding identifier. The directory system, in turn, routes the user to the metadata source associated with the globally unique identifier for the content object.

#### Metadata Directory Supporting Content Objects with Multiple Content Identifiers

[0038] Another novel method addresses content objects with two or more content identifiers, potentially referencing different metadata sources. This method registers different globally unique identifiers for a content object. These globally unique identifiers each comprise a content identifier provided with the content object and a bounding identifier identifying a set of content identifiers of which the content identifier is a member. For each of the globally unique identifiers, information is maintained about a metadata source. The method receives a first content identifier for the content object, and uses a bounding identifier associated with the set of the first content identifier to determine the globally unique identifier for the first content identifier. The user is routed to the metadata source associated with globally unique identifier.

[0039] This approach handles a variety of cases in which two or more content identifiers are provided for a content object for the purpose of registration or resolution. The metadata directory system supports and manages both the registration of and routing to different metadata sources corresponding to different content identifiers of the content object.

#### These Cases Include:

[0040] 1. Content identifiers are embedded or calculated by different ID providers and are later derived from the content object using different readers associated with those technologies. For example, the readers are different because they derive the content identifier using different content identification methods (e.g., through the file header/footer, digital watermark, fingerprint, Vertical Blanking Interval data in video programming, etc).

[0041] 2. The different readers may, for example, derive the content identifiers using different attributes of the content object. These different attributes may comprise different types of embedded auxiliary data (different watermark embedders/readers, watermark vs. embedded header/footer data). These different attributes may comprise attributes from which different digital watermarks or robust hashes are derived. The different attributes may correspond to in band and out of band attributes of the content object. "In band" refers to an identifier derived from content in the content object that is rendered for perception by a human. "Out of band" refers to auxiliary data carried in the content object but not forming part of the content that is rendered for perception by a human. Certain types of content objects include multiple content programs rendered for perception by a human, like video and audio tracks and close captioned text. In band identifiers may be derived from one or more of these content signals within the content object. In some cases, one content program may be embedded in another content program within a single content object, such as where close captioned text is embedded in the audio or video program of an audiovisual work.

[0042] 3. The different content identifiers for a content object may be derived from the content object using different parts of the content object, including different in band and out of band parts as well as different parts within the in band portion of the object and different parts within the out of band portion. These parts may be in discrete locations in one domain of the content signal, yet at overlapping locations in others. Examples of domains include spatial, temporal and transform domains (e.g., frequency domain, compressed domain, etc.) of the content signal in a content object).

#### Enabling Different ID Provider and Content Provider Participants

[0043] In some metadata systems, the system owner, serving as a registration authority (RA), provides the identification technology and content owners use the technology to register themselves as a content provider, register content and link the content to metadata.

[0044] This document describes a novel system that enables multiple identity providers (ID Providers) to register and use the system. The ID Provider registers with a metadata directory system, receives a unique bounding identifier, and uses this bounding ID (e.g., an ID provider ID) with subsequent interactions with the metadata directory system. Separately, metadata source providers register meta-

US 2007/0156726 A1

Jul. 5, 2007

5

data sources with the metadata directory system. This enables many different participants to associate content objects with metadata sources using one or more identify providers. Examples of metadata source providers include content providers, like content owners or retailers that have the flexibility of working with different ID providers to associate content objects with metadata. Both content providers and ID providers can register and use the system. The metadata source is the system or device that provides the metadata, like a web site. The directory system uses an identifier for the metadata source, which enables it to maintain an association between a content object and its corresponding metadata source. For example, in some embodiments, a URL serves to identify the location of the source.

[0045] One embodiment of the directory system is referred to as CMDS. CMDS enables content providers to utilize the CMDS to knit together metadata sources that are associated with content using disparate and previously incompatible ID provider technologies. CMDS enables content providers to manage their proprietary information (i.e. they do not have to turn over control of proprietary metadata to a RA for storing and distributing the metadata), enables eCommerce for all value chain participants (e.g., both content owners and retailers can embed CIDs), facilitates interoperability with all content identity provider technology (even pre-existing ID systems, such as EPC), allows for compatibility with both PC and mobile devices, facilitates interoperability for multiple ID providers who license a common identification algorithm, and enables usage reporting and vital marketing statistics.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0046] FIG. 1 is a flow diagram illustrating a method of associating a content object with a metadata source.

[0047] FIG. 2 is a flow diagram illustrating interaction between a reader and a directory system to link a content object with a metadata source.

[0048] FIG. 3 is a system diagram illustrating a metadata directory system and its interaction with ID providers for registration of content IDs and readers for resolution of content IDs.

[0049] FIG. 4 is a system diagram illustrating an overview of a content identification and routing system.

[0050] FIG. 5 is a diagram demonstrating a usage model of using the routing system to direct a user to content providers based on content identifiers extracted from the user's content object.

[0051] FIG. 6 is a diagram illustrating an implementation of a router in more detail.

[0052] FIG. 7 is a diagram illustrating a distributed router system in one implementation of a directory system.

[0053] FIG. 8 is a diagram illustrating an example of a directory system architecture along with different content owner participants using the system to associate content identifiers with metadata sources they control.

[0054] FIG. 9 is a diagram illustrating a routing system including a rules processor and traffic monitor.

#### DETAILED DESCRIPTION

[0055] FIG. 1 is a flow diagram illustrating a method of associating a content object with a metadata source. This diagram is intended to show how a content metadata directory system performs registration of content IDs, which achieves two objectives:

[0056] 1. it enables integration of different content identification schemes with potentially overlapping content ID schema; and

[0057] 2. associates the each content ID with one or more metadata sources.

[0058] As shown in block 100, the directory system receives content IDs. These content IDs either originate from pre-existing sets (e.g., pre-assigned by an ID provider), or directory system itself issues content IDs for an ID provider upon request. The ID provider refers to an entity that provides content identification for content objects. Typically, this is a content identification technology provider, such as a provider technology for computing in band (e.g., watermarking or fingerprinting) or out of band identifiers (DRM container technologies, VBI inserters, etc.) for content objects.

[0059] To differentiate among different sets of content IDs, the directory system provides a unique bounding identifier (BI) as shown in block 102 for each unique set of content IDs registered in the directory. It ensures that BI's for different sets of content IDs do not collide. Thus, while it is possible to register pre-existing BI's that do not collide, it is preferable for the directory system to issue the BI's or at least issue guidelines for their use to prevent collisions.

[0060] The directory system forms globally unique identifier (GUI) for all content objects that it manages by combining the content ID for an object with the BI for the set of content IDs of which the content ID in question is a member (block 104).

[0061] The directory system associates the GUI for a content object with a metadata source as shown in block 106. As explained below, this metadata source provides metadata in response to a request from an entity that supplies the content ID for a content object. The directory system stores the association between the GUI of a content object and the metadata source in a manner that enables fast, efficient routing of the requesting entity to the metadata source. In one implementation, the directory system stores a location of the metadata source on a network, such as a URL. This enables the requesting entity to connect to the metadata source and retrieve metadata associated with the content object. Several metadata sources may be associated with a GUI and returned to a requesting entity.

[0062] FIG. 2 is a flow diagram illustrating interaction between a reader and a directory system to link a content object with a source of metadata. As shown in block 110 the reader extracts a content identifier from a content object. It then forwards the content ID to the directory system and either implicitly or explicitly identifies the BI for the set of content IDs in which the extracted content ID is a member (112). An example of explicitly identifying the BI is where the reader supplies a unique provider identifier assigned to it, along with a version identifier. The version identifier may be used to enable the provider of the identification technol-

US 2007/0156726 A1

Jul. 5, 2007

6

ogy to create different content ID sets, which enables flexibility with different versions and upgrades of the ID provider's technology. Some examples include different versions of a particular RFID, bar code, digital watermarking, fingerprinting or DRM technology for different customers, applications or content object groupings.

[0063] An example of implicitly identifying the BI is where the directory system determines the identity of the provider based on some inherent attribute of the data provided by the reader, such as its format, data type, etc.

[0064] The directory system forms the GUI using the received content ID from the reader and the BI (114). The precise manner of formation may vary. One approach is to concatenate the content ID and BI in some fashion (e.g., appending them together to create one GUI). Another approach is to hash one or more parts of the content ID and BI and combine these parts to create one GUI. It is possible that one or more third parties may be involved in the process of supplying the content ID and BI parts of the GUI. For example, a fingerprinting database may supply the content ID and the BI associated with it. Yet another approach is that each ID is located in a separate field in the database, and only entries that match all ID fields are used to register or resolve the GUI. The directory system maps the content ID and BI into a naming, numbering or address space which provides a GUI for each content object.

[0065] The directory system then looks up the metadata source or sources associated with GUI (116) and returns some identification of the source to the reader. One form of source identification is its URL, but other forms of identification are possible as well. Examples include a pointer, address, index to a database of metadata, machine instruction for accessing the source. The approach of returning an identification or location of the source to the reader enables the reader to establish a direct connection with the source to get the metadata. In alternative approaches, the directory system can instruct the source to establish a connection with the reader by providing the reader's address. The directory system may also act as an intermediary between the reader (or location specified by the user of the reader), on the one hand, and the metadata source on the other. This approach can be used where the directory, or some other intermediary in communication with it, is used to provide additional processing of the metadata before supplying it to the requesting entity at a location of the requesting entity's choosing.

[0066] FIG. 3 is a system diagram illustrating a metadata directory system and its interaction with ID providers for registration of content IDs and readers for resolution of content IDs. As shown, different ID provider systems 200, 202 establish sets of content IDs corresponding to content objects. The ID provider systems may pre-assign sets of content IDs to objects and then register with the directory system 204 them via the directory system's registration interface 206. Or, they may request the directory system 204 issue content IDs via the registration interface 206.

[0067] The directory system includes and/or communicates with a database 208 that stores the association of the GUI 210 for each content object and its corresponding metadata source ID 212. FIG. 3 shows a generalization of the association of a GUI to a corresponding metadata source. This database can be structured with one or more layers of

indirection in which aspects of the GUI are used to index into different handlers and/or databases segmented by provider, version, location, etc.

[0068] FIG. 3 illustrates the point that different ID providers (e.g., providers A and B) or different versions of a single provider's technology (Versions A and B) can register multiple sets of content IDs with the directory system. Moreover, multiple content IDs from different sets may be associated with the same content object 214. For example, ID provider A and ID provider B distribute readers A and B (216, 218), respectively. These readers may be programs, devices, or a combination thereof (including components of a device or distributed over different devices, such as in a web services type implementation). In this depiction, content object 214 includes CID 1 of set A (220) and CID 1 of set B (222). A single program or device such as a media player program or device may include both readers A and B and present a common graphical user interface to the user. Alternatively, the readers may be separate components or programs executing within the user's device or network domain.

[0069] When the readers 216, 218 encounter the object, such as upon file open/copy/transfer/edit command, entry into a device, network gateway or filter, they extract the content ID and forward it (along with the implicit or explicit BI data) to the directory system's resolution interface 224. The resolution interface, in response, looks up the metadata source ID or IDs if there are multiple sources (e.g., a URL or set of URLs) and returns them to the respective readers (or some program or device designated by the readers or pre-registered with the directory system).

[0070] The readers 216, 218 then use the metadata source IDs to establish connections with the respective metadata source provider systems 230, 232 and get the metadata associated with the content object 214 from these different sources. For example, these sources may be web content files on web servers located at the location returned by the directory system. These sources may be maintained or controlled by different participants in the content distribution value chain, such as content owners, retailers, catalog companies, product manufactures, service providers, etc. One source of metadata may be the content owner, which provides web content affiliated with the content owner, and another source of metadata may be the content retailer, which provides web content affiliated with the retailer (e.g., such as eCommerce opportunities to buy related entertainment content or merchandise).

[0071] The specification provides detailed embodiments of technology summarized above as well as additional inventive systems and methods. FIGS. 4-8 are briefly summarized here, are further illustrated with examples in connection with a system called CMDS.

[0072] FIG. 4 is a system diagram illustrating an overview of a content identification and routing system.

[0073] FIG. 5 is a diagram demonstrating a usage model of using the routing system to direct a user to content providers based on content identifiers extracted from the user's content object.

[0074] FIG. 6 is a diagram illustrating an implementation of a router in more detail.



US 2007/0156726 A1

Jul. 5, 2007

7

[0075] FIG. 7 is a diagram illustrating a distributed router system in one implementation of a directory system.

[0076] FIG. 8 is a diagram illustrating an example of a directory system architecture along with different content owner participants using the system to associate content identifiers with metadata sources they control.

[0077] As summarized above, the system works with many different content identification technologies for both electronic and physical objects. It also applies for both out of band and in band content identification. Examples of out of band content identification for content objects include identification based on auxiliary data in file headers and footers, Vertical Blanking Interval (VBI) inserted data, DRM container schemes for identifying content signals in the container, etc.

[0078] The following describes some examples of in band content identification in more detail.

#### Digital Watermarking

[0079] One method for in-band content identification is to carry a content identifier in a digital watermark embedded in the perceptual portion of a content object that is rendered for display or playback to a human. The digital watermark is hidden or “steganographically embedded” in the content by modifying the content to include an auxiliary signal that conveys auxiliary data (e.g., a message “payload”), such as the content identifier.

[0080] Digital watermarking is a process for modifying physical or electronic media to embed a hidden machine-readable code into the media. The media may be modified such that the embedded code is imperceptible or nearly imperceptible to the user, yet may be detected through an automated detection process. Most commonly, digital watermarking is applied to media signals such as images, audio signals, and video signals. However, it may also be applied to other types of media objects, including documents (e.g., through line, word or character shifting), software, multi-dimensional graphics models, and surface textures of objects.

[0081] Digital watermarking systems typically have two primary components: an encoder that embeds the watermark in a host media signal, and a decoder that detects and reads the embedded watermark from a signal suspected of containing a watermark (a suspect signal). The encoder embeds a watermark by subtly altering the host media signal. The reading component analyzes a suspect signal to detect whether a watermark is present. In applications where the watermark encodes information, the reader extracts this information from the detected watermark.

[0082] Several particular watermarking techniques have been developed. The reader is presumed to be familiar with the literature in this field. Particular techniques for embedding and detecting imperceptible watermarks in media signals are detailed in the assignee’s U.S. Pat. 6,122,403 and 6,614,914, which are hereby incorporated by reference.

#### Robust Content Hashes and Fingerprinting

[0083] Another method of in-band content identification is a hash of the content data, which is sometimes referred to as a content “fingerprint.” In order to remain unchanged through distortion of the content, a robust form of hash is

sometimes used in which the hash is derived from features of the content that are expected to survive in tact through distortion in the delivery channel, like clipping, time or geometric changes, compression, transmission, etc. Examples of these features include a vector of frequency domain values (e.g., e.g., robust low and mid frequencies), perceptually important features (including temporal, spatial or frequency domain features), content statistics, feature values quantized into discrete bins, all of which are not necessarily mutually exclusive, and which are generally characterized in that they are not degraded by expected distortion in the channel (e.g., compression, transmission, D to A, and A to D conversion). To be clear, robust fingerprinting methods allow some change in the content signal, yet the fingerprint computed from that distorted signal still maps to the same identifier. In other words, expected degradation does not change the signal so substantially that it maps to a different fingerprint in the database or no fingerprint at all. For consistency, we refer to these methods as fingerprinting, which generate content fingerprints.

[0084] This fingerprinting approach to content identification has the advantage that the auxiliary data embedding process is unnecessary. Instead, the reader process can generate the identifier from the content object without prior explicit modification of the content object to include auxiliary identifying data. A potential disadvantage is that copies of the same content program (e.g., the same musical track, song, movie) have the same fingerprint, which requires use of additional means to differentiate different copies of the same content program. The advantage of not requiring auxiliary embedding is also mitigated by the fact that the fingerprint needs to be registered and kept in a fingerprint database to enable matching of a computed fingerprint with registered fingerprints. Once a match is found, the database provides the content identifier for the matching fingerprint. This potentially adds additional processing and network communication to produce the content identifier.

[0085] For ease of understanding in the context of our architectures, we describe fingerprint methods as including three components, a calculator, a reader and a fingerprint database. The calculator does the following: (1) creates the fingerprint using the same (or similar, where changes are based upon known or estimated distortion) algorithm as the reader, (2) registers the fingerprint in the fingerprint database, and (3) links the fingerprint to a content identifier. The fingerprint may be one value or a set of sub-fingerprints taken from portions of content throughout some or all of the content. When sub-fingerprints are used, each sub-fingerprint or set of sub-fingerprints links to the same content identifier. The reader computes a fingerprint (e.g., set of sub-fingerprints), sends them to a fingerprint database, and receives the content identifier.

[0086] The fingerprint algorithm, as used in the calculator and reader, utilizes the perceptual content signal. The fingerprint is a numerical construct (e.g., an array of values) derived from a content signal that serves as a statistically unique identifier of that signal, meaning that there is a high probability that the fingerprint was derived from the content signal in question rather than a different one that sounds or looks similar. One component of fingerprint algorithm is a form of hash algorithm. The hash algorithm may be applied to a selected portion of a content signal (e.g., the first 10 seconds) to create a fingerprint, or may be applied repeatedly

US 2007/0156726 A1

Jul. 5, 2007

8

to generate a sequence of robust hashes, where a specified sub-set of this sequence can identify content. For example, the sequence may use sub-fingerprints from every  $\frac{1}{16}$  second of a song, and require 32 sub-fingerprints (i.e. any 2 seconds of audio) to identify the audio. In addition, 3 seconds can be used to improve accuracy.

#### Directory System Applications

[0087] As noted, the directory system and method are applicable to both electronic and physical content objects (as well as objects that pass to and from analog and digital domains). The network methods for communicating and routing a device (such as computer or wireless phone handset) having a content object to another having metadata relating to the content object apply to different types of networks, including computer networks like the Internet, and wireless telephone networks. Examples of mobile device applications, such as linking from an object to its metadata via a cell phone handset, are described below. The router can be implemented in the cell phone network, the Internet, or spanning both the cell phone and Internet, such as when mirror routers reside in the cell phone network, the Internet or both the cell phone and Internet network. Different types of URLs such as WAP, WMI and “full” may be used as metadata source identifiers maintained in the directory system.

#### Metadata Routing and Rules

[0088] A content metadata directory system enables those having control over a content object to associate a metadata response with the content. In many applications, it is advantageous to allow different metadata responses for a particular content object. We refer to the instructions and/or data within the system that control the metadata response as “rules.” One example of a rule in the directory system is data or instructions specifying the URL or list of URLs to return in response to a metadata request specifying a particular content ID. In some cases, more complex rules are needed to support metadata responses tailored to a particular context, such as user attributes (e.g., user preferences, security, account information and status, location, etc.) or transaction attributes.

[0089] To illustrate an implementation of these rules, it is instructive to begin with a diagram showing the components of a router system in which these rules operate. FIG. 9 is a block diagram illustrating a router system and its relationship with a consumer and content distribution participants. In this diagram, there are at least three entities represented: the consumer, the router service provider(s) and content distribution participants. FIG. 9 shows a configuration in which the components of the router system are distributed across these participants. The components comprise devices and/or software modules and examples of these are provided throughout this document. One set of components are referred to as part of the consumer domain 900. These are components that the consumer controls such as his or her devices and embedded software in these devices (such as wireless phone, home network, PC, home entertainment system devices, etc.). Another set are referred to as the router system 902. These are components involved in routing from content identification and related context to a metadata response, as well as related network services, such as metadata traffic monitoring and response aggregators. A final set are referred to as the content distribution partici-

pants 904. These are the participants that provide content objects, manage their distribution, and supply and/or control metadata content distribution.

[0090] The content distribution participants 904 include, but are not limited to, content owners 906 and content distributors 908. These owners include not only traditional providers of entertainment content, but also those that provide content, including advertising, used to market other products and services. These participants register their content objects 910 with the router system 902 and distribute them to consumers 912. Through a registration component 914, the participants 904 register their content objects in the routing system as described in the various embodiments in this document. In particular, they provide content identifiers and associated rules for providing responses to metadata requests for these identifiers. These content identifiers and rules are stored in an ID registry database 914 and rule database 918, respectively. These databases may be integrated in one database or implemented in separate databases with a key or like identifier (e.g., unique identifier) that associates a content object with a corresponding rule governing the metadata response.

[0091] The router system enables consumers to request a metadata response wherever the content object travels in its distribution chain as long as the consumer has a reader to extract a content identifier and associated modules for packaging and sending a metadata request to the router system. FIG. 9 illustrates the metadata request as an “ID labeled request” from the consumer domain. The “consumer domain” in this context is not intended to be limited to only retail users of content. Rather, it includes virtually any user of the router system whether they represent content owners or distributors seeking to access metadata for their content, purchasers/licensees of the content, or customers that receive and consume content for entertainment, information or commerce. In fact, rules help facilitate different levels of access among the various types of consumers of the router system.

[0092] The router system includes an ID resolver 920, which receives the ID labeled request and determines an ID for the content object based on the content ID and associated identification system information supplied by the requesting device. The discussion for mapping content identifiers and provider identifiers into a unique identifier for a content object is one example of such a process. This approach of unifying content identification schemes enables the routing system to serve as a form of lingua franca for content identification, stitching together disparate content naming schemes.

[0093] The content ID may have one or more metadata responses registered for it. A rules processor 922 executes rules associated with the content object (e.g., via the content ID) to determine the metadata response based on information supplied in the request, information about the requesting user (including the user’s device capabilities, connectivity, etc.), and/or information retained in the routing system, such as transaction data for the content object or the class from which it the content object originates (class defined by content genre, by affinity group of consumers for the content genre, etc.).

[0094] In addition to resolving IDs and executing rules for content objects, the routing system may also provide addi-

US 2007/0156726 A1

Jul. 5, 2007

9

tional services. The router system of FIG. 9 includes a traffic monitor 924, which logs usage statistics and generates usage reports. An embodiment of this usage monitoring and reporting is described with reference to CMDS implementations below.

[0095] The rules processor can also execute rules based on a content object's dynamic metadata. Dynamic metadata refers to metadata that change over time. One example of such dynamic metadata is a content object's usage data. A rule governing the metadata response can be made dependent on the usage data. For example, if the number of requests for metadata exceed a threshold, the metadata response is adapted accordingly. For example, more metadata data including information about related content and commerce opportunities are provided as the interest level in a particular object increases over time. The response can be tailored based on user information derived from metadata requesters so that the response is tailored to the common attributes of the class of users requesting metadata for the object. Patterns of common attributes of users, content or the content metadata emerge as usage data is tracked over time. This enables the system to identify and dynamically add metadata responses for a content object. For example, as interest in the content object grows, the routing system adds additional links to related content objects and products and services to an affinity group for a content object or class of object. The affinity group can be defined, for example, as a group with common preferences or interests in content objects.

[0096] As a further service, the routing system can also act as a metadata repository and aggregator of metadata responses. FIG. 9 includes a response aggregator 926, which provides metadata responses and stores metadata for content objects in support of this function. In some embodiments, the routing system simply re-directs the metadata request to the content owner/distributor, which in turn, provides metadata to the requesting consumer. There are alternative ways to implement this approach. One, as documented in CMDS embodiments, is to return to the consumer a URL or set of URLs for a metadata sources controlled by others. The consumer's device (e.g., via a web browser or like application program) then sends a request for metadata to the metadata repository at this URL. Another approach is to act as a proxy server for the content owner's metadata repository. In this case, the routing system determines the metadata source's URL based on ID resolution and rule execution (if rules exist) and issues a metadata request to the metadata repository at this URL. If multiple URLs are involved, it makes a query to each one. The routing services aggregate the metadata responses from the metadata repositories and returns the aggregated metadata to the requesting consumer. Another approach is where the routing system forwards the metadata request from a consumer to the URL of a metadata repository, which in turn, responds with metadata directly to the requesting consumer. In sum, the content owner/distributor controls the metadata response in some cases as shown in block 930, where as the routing system controls the response in the other case.

[0097] As an added function, the routing service acts as a metadata repository. In cases where a rule dictates it, a user requests it or other metadata sources are not available, the routing system identifies the URL of the metadata repository

within its control, packages the metadata response with the response aggregator and returns the metadata to the requesting consumer.

[0098] The capability of serving as a metadata repository enables the routing system to provide additional network services. One such service is to enable users the ability to collaborate in the creation of metadata for content objects by posting recommendations, preferences, and other related information about content objects in the metadata repository. This adds an added dimension to affinity groups identified and managed by the routing system. In particular, it enables members of these groups to be active contributors to the metadata for the content objects that they are most interested in.

[0099] Having described the routing system, we now describe rules implementations in more detail. One approach to supporting multiple different rules for a content object is to enable one or more participants that control the object to register different content IDs associated with that object in the ID registry, each having an associated URL or set of URLs in the registry database. For example, each content ID references a URL for the respective participant (e.g., content owner, distributor, retailer, each providing different metadata sources at the corresponding URLs in the database).

[0100] This approach may lead to further rules being implemented in the router system and/or client program executing in the consumer's device. For example, the process begins with a client reader program or programs on the consumer's device extracting multiple CIDs from the content object and forwarding them to the router system. The router system, in turn, looks up the corresponding URL or URL list for each CID. Then, the router executes a rule governing which CID or CIDs have priority and returns a subset of the URLs associated with the CIDs with priority. Alternatively, the router refrains from prioritizing the requests and returns all URLs associated with each content ID to a client reader program in the consumer's device. The client program either displays a web page with hyperlinks to each URL, or it executes rules to select which URL or set of URLs has priority, and then re-directs (e.g., sends a metadata request) to the URL with the highest priority.

[0101] A single CID may be associated with metadata responses from several different participants (search engine provider, metadata aggregator, distributor like iTunes, Record Label, advertiser, etc.). In this case, the registrant of the CID specifies the URL or URL list corresponding to each of the metadata responses for that CID. In addition, the registrant specifies a rule governing the conditions in which the different metadata responses are triggered. The registration process is facilitated by a graphical user interface accessible via a network, such as a web interface enabling the registrant to list for each metadata response:

[0102] 1. the URL or set of URLs for the desired metadata response;

[0103] 2. the conditions that cause the routing system to select the metadata response.

[0104] The conditions are a function of the attributes of the routing transaction. These attributes fall into two categories: user attributes and non-user attributes. The user attributes are obtained by the router through the user registration process, in data provided in the metadata request,



US 2007/0156726 A1

Jul. 5, 2007

10

and/or in data derived by the router based on history of requests from the user. The user can specify metadata preferences through registration or in the metadata request. Preferences can include likes/dislikes about content genres, likes/dislikes for product/service advertising, preferences in content format, preferences for types of media players and media player settings, device capabilities, etc. Non-user attributes include attributes about the transaction derived by the system, such as the time, geographic territory, history of transactions relating to the content object or content objects in the same genre as the content object, type of object, etc. For example, the router's usage data provides information about the popularity of the content object, correlation to the preferences of others who have requested metadata for the object, etc. In some applications, the reader packages information about the object along with the extracted content ID, such as the object type and format. This enables the metadata response to be tailored to the object type and format that the user has the capability to render on his or her device.

[0105] A typical rule registration interface enables the registrant to select different URLs depending on the attributes of the routing transaction. For example, rule 1 dictates: use URL 1 when artist preference=TRUE; Account status=active subscriber; and Advertising Tolerance=LOW.

[0106] Rule processors can be implemented within and executed within one or more different locations along the metadata request and response paths. Before enumerating these locations, let's review a summary of the locations in a CMDS embodiment. The first location is the requesting client, which is within the consumer domain shown in FIG. 9. The next is the router system, which itself, has different components that can contribute to rule processing. For simplicity, FIG. 9 shows the rule processor as a single block, yet the rules processing function may be distributed. The next location along the path is either back to the requesting client, or to metadata source to which the request has been re-directed by the router.

[0107] At content identification and metadata request, rules are executed to control:

[0108] 1. what type of CID the client extracts (is client going to trigger fingerprint based CID retrieval, DWM CID retrieval or header/footer CID retrieval?)

[0109] 2. what client sends to the router (one or multiple CIDs)

[0110] At the router, the rules processor executes a rule or rules to control:

[0111] 1. Which CIDs, if multiple CIDs for one content object are provided, has priority?

[0112] 2. Which URLs the router returns to client (e.g., single/multiple URL per each CID, or does one CID dominate?)

[0113] 3. Whether the registrant of a content object has requested that the request be re-directed to a URL of a device it controls, which in turn, provides a metadata response to the client.

[0114] At the client, upon response from the router or other device per above, the rules are executed to control:

[0115] 1. what client displays (are URLs prioritized or not?)

[0116] 2. what client sends to web server in response to packet returned by router.

[0117] At the client, a client program used to consume or manage content is equipped with a reader for one or a few types of content IDs. This client controls what IDs get sent to the router and thus, controls which entities are linked to.

[0118] Alternatively, there are multiple readers, each acting independently according to their own rules.

[0119] Alternatively, the router sends all URLs for metadata sources associated with corresponding CIDs back to the client and the client decides which one to use. The client may also determine what type of context data it provides to the metadata source, such as GPS information, depending on whether the user wants to get or has paid for location based services.

[0120] To respect user privacy, user preferences can be maintained solely at the client. The client maintains control over whether and when the user preferences and attributes are forwarded to the router and/or metadata source/repository. In some cases, they are not provided at all, and the client aggregates and then tailors the metadata response from multiple URLs based on user preferences. In other cases, the client forwards the preferences to the metadata source directly after receiving its URL to the router. In this case, the router does not get access to the user preferences.

[0121] At the router, the router determines based on context or other information from the client how to re-direct the client to a metadata source located at a URL.

[0122] The metadata source at the URL sends different types of metadata or links to a requesting client, which enables the client to pick or the user to pick from among metadata/links presented in a user interface of the client.

[0123] As noted above, the metadata responses may be prioritized based on user preferences. In such embodiment, the client is programmed to get or learn user preferences and prioritizes links to metadata returned by the system accordingly.

[0124] In another embodiment, the router system provides additional services for managing users, including, for example, authenticating users and managing user accounts. This can be explicit management of information supplied by the user, or management of user transactions based on consent provided by the user. In either case, the router derives a user classification based on information it has gathered about the user's preferences for consuming content. In one embodiment, the router system classifies routing transactions based on the user's willingness to pay for products and services. Those willing to pay for premium content get a metadata response with opportunities to access this premium content, while those not willing to pay get no fee metadata responses that are subsidized by advertising, for example. Further, if the router has authenticated the user and the user's account status, it re-directs the user to URLs that are secure electronic commerce sites, initialized based on the user's identity. This enables the routing system to link the user directly to electronic transactions without requiring the metadata source to handle the authentication processing.

[0125] One particular example of the authentication service in the router is to enable direct linking into a Digital Rights Management (DRM) server or other e-commerce

US 2007/0156726 A1

Jul. 5, 2007

11

server, in a state where the client is pre-authenticated. The client and router provide authentication information needed to complete an electronic purchase through private and/or encrypted data fields, or a combination of public and private fields, where private only readable by certain entities.

[0126] Another service of the router is track the flow of content object sharing over networks. Digital certificates, or other identifying information of users, is used to detect different users that request metadata for a particular content object, which is uniquely identified via content identifier. This tracking of content object flow by content ID and user certificates provides data to the content owner regarding how content objects flow through networks of users. This provides another means to identify an affinity group for content and tailor content and metadata distribution to the preferences of the affinity group. This tracing method traces content objects as they are processed by new and different users. Each time a request for metadata is received, the router logs the request and also records user identification for the request, if available. The router analyzes this log to identify the path of the content object through new users.

[0127] As demonstrated in the example above, the router enables metadata responses to be governed by a hierarchy of rules distributed across the system, including macro rules implemented in the router that specify URLs to return, and micro rules implemented in the client that further control how the client presents and links to these URLs.

[0128] Examples of micro rules include: rules governing how authentication of a user occurs to enable access to different type of metadata sources, (e.g., use of identity triangle—what user knows—password, has—access token, ID card, and is: biometric login and user verification). The level of authentication dictates the nature of the metadata provided (what links, metadata, etc. are provided). The level of authentication also dictates purchase limits and usage rights (rights for redistribution or sharing with other users).

[0129] An additional service of the router system is the support for handling private and public fields in the data packets sent as metadata requests to the router. For example, the client device in the consumer domain sends public and private fields (public field is generally readable, private field is packaged by client and likely encrypted, intended to be read only by authorized service, such as secure transaction server that the router links to). The router uses the public field, for example, as data input for rules determining the URLs for the metadata responses. The metadata sources decrypt and use the data in the private fields to provide tailored information to the user, while keeping the user's information private and secure.

[0130] The rules in some systems also govern the amount of network resources allocated for a metadata transaction. In particular, the size of the socket, data pipe or channel opened between the metadata source and the client is a function of client's authentication level and user account status (e.g., indicating willingness and ability to pay).

[0131] There are a number of ways to propagate rules through the network infrastructure. In one embodiment, rules are pushed to client through content auxiliary channels in the content objects (such as file header/footer, encryption container, digital watermark, etc.) In another embodiment, rules are distributed with media consumption programs,

such as media players, like Windows Media from Microsoft Corporation and iTunes from Apple Computer, etc.

[0132] Each component (client, router, and metadata source) executes its own set of rules established based on what it has learned about the user.

[0133] As noted above, the input to the rules includes user and non-user attributes, including, but not limited to user demographics, device platform, age, geography (GPS), time, search engine metrics that prioritize results for content searches based on the user's search history, which indicates preferences for types of content.

[0134] The input to the rules includes preferences derived from user behavior, content consumption history, most frequently accessed content, etc.

[0135] Another input includes content type, which enables the content to be targeted at content type preferences of the user or the user's rendering equipment. Also, the rules specify different metadata responses based on whether the client device is operating in on line or off line mode. Offline mode instructs the client to re-direct a metadata request to a metadata source in its cache of metadata, which serves as a local metadata repository.

[0136] The router system provides a number of opportunities for revenue generating data services. One such service is revenue sharing model in which fees collected for distributing metadata are shared with content owners as a function of usage data. In an embodiment of this service, the router tracks usage data corresponding to metadata requests for content objects and the usage data for these content objects is used to determine revenue sharing among content owners and distributors. The router tracks usage data and this usage data is used to determine how artists that created the content get a share of fee collected under a license in which collected fees distributed to the artists based on metrics dependent on the distribution of metadata requests for the content objects. Additional models of revenue sharing are fee based, where a portion of the fees paid by consumers for metadata responses are distributed to the content owners.

[0137] Another service of the router is an auditing function. One such audit is checking the validity of URLs supplied to it during registration. The router periodically checks URLs in its database to make sure they are current, valid, responsive, etc. The router also provides metadata request trend analysis that enables the metadata sources to anticipate metadata request work load and allocate more resources to serve metadata based on the anticipated workload.

[0138] The router itself is distributed and has instances of itself that are mirrored across the network. The router specifies to the clients which among several URLs to use for subsequent request for routing services.

[0139] Another service of the router is to track the form of content identification used to identify the content object for each transaction. This enables the router to flag content for subsequent content labeling, including layers of different content IDs. For example, a content object identified using a fingerprint is flagged for embedding a digital watermark. The digital watermark provides a finer grain identification than the fingerprint by differentiating among copies of the

US 2007/0156726 A1

Jul. 5, 2007

12

same content. The content ID in the watermark is then registered and associated with metadata responses that are tailored to a particular copy of the content object.

#### More on User Generated Metadata and Rules

[0140] Above, we described how to incorporate user generated metadata into the metadata routing system. The routing system has the flexibility to incorporate its own metadata repository for user created metadata, to link to user generated metadata repositories maintained by others, and to integrate a combination of both types of metadata sources. By establishing an interface that allows metadata providers to register links between content and metadata sources, the routing system has the flexibility to integrate different metadata sources into one unified metadata service. For example, the routing system maintains links between the identifier (or identifiers) for a piece of content (e.g., song, TV or radio program, movie, Podcast, advertisement, etc.) and sources of metadata, including user generated metadata within or outside the routing system's domain. The database, in particular, stores the unique identifier of the piece of content and associated URLs of the user generated metadata for that piece of content.

[0141] In particular, the routing system can specifically link to the user generated metadata maintained in different user generated metadata sites on the Internet. Examples of web sites that have implemented systems for user generated metadata include Flickr and Del.icio.us, which provide web services operated by Yahoo. These systems enable users to apply metadata, called "tags," to pieces of content. In particular, Flickr allows users to upload images and add tags to images within the context of the Flickr system. However, the tags are not persistently linked to content, and therefore, as content moves outside the bounds of the Flickr system (e.g., the Flickr servers or connected client device), the tags are easily lost. The routing system provides a means for persistently maintaining links between a content object and its metadata, even as the object is distributed across different domains, devices and networks. Further, it allows a content object to be persistently linked to several different user generated metadata sites.

[0142] When these capabilities of persistent linking across different sources of metadata are combined with rules based processing, the system fulfills a more sophisticated array of metadata requests across a broad array of metadata sources. Each piece of content becomes its own node or portal interconnected with disparate sources of metadata across heterogeneous distribution methods, networks, content formats, and devices. In addition, rules processing in the system filters metadata according to user preferences.

[0143] It is anticipated that in many applications, users will prefer user-community generated metadata over the original metadata from the content owner or distributor. An example is user generated metadata that assigns content ratings to content for parental control. In particular, a user may trust the content rating applied by the trusted community of which he or she is a member over the rating applied by an industry group. Today, as long as the user stays within the domain of a provider of trusted content (e.g., within a connected session to a web service), the user can take advantage of the metadata associated with the content in that domain. However, when content travels outside the domain (e.g., is stored in a device that does not respect or even

understand the protocol or metadata from the domain, goes offline, is trans-coded or played, is emailed to a friend), then a scheme is needed to re-associate metadata and apply the user's preferences. The routing system supports this re-association and enforcement of user preferences by allowing the user to request metadata with the user's rules that specify that, in certain contexts, the content rating in the metadata from the user group is to be used instead of the industry content rating.

[0144] A typical usage scenario of the system to apply ratings is as follows: user encounters an content (in any of a myriad of ways: e.g., receives from a friend, downloads from a social networking site, records from a live broadcast, searches an archive); the user's device includes an application that incorporates the reader, which extracts a content ID from the content (or alternatively, the user selects the content, which triggers transmission of it to a remote reader that does the same); the reader forwards the content ID along with user preferences to the routing system, the routing system extracts the related metadata from linked sites specified by the associated URLs, the routing system applies the rules applicable to the content, including those based on user preferences, and sends back a metadata response to the user that informs the user of the rating and/or the reader automatically enforces the rating by controlling play out of the content.

[0145] These metadata sources, as noted, can be dynamic metadata sources based on dynamically generated metadata from searches, RSS feeds, and mash-ups. The URLs identify more than just IP addresses of physical devices, and extend to dynamic metadata sources in virtual environments such as Second Life (URL in protocol for Second Life adheres to SLTP protocol).

#### Federated Content Identity

[0146] The concept of federated identity has emerged in response to demands for a user-friendly and interoperable framework for establishing user identity across disparate domains (e.g., across the security walls erected around different entities' databases, such as the metadata services of a content distributor, content owner, social networking site, metadata tagging sites like Flickr and del.icio.us). Simply stated from the user perspective, "federated identity" is an identity management framework that enables the user to access disparate data services, each with their unique secure log in procedures, with a single log in. As described below, embodiments of the routing system leverage federated identity technologies, such as SAML and WS-Trust, to establish and enforce user identity for users of the routing system, including both providers and consumers of metadata.

[0147] The disparity of content identification and metadata protocols creates a similar demand for a scheme of federated content identity. The routing system satisfies demand by creating a framework that unifies disparate content identification technologies (such as digital watermarking, content fingerprinting, headers in files whether un-encrypted or part of an encryption container) as well as incompatible protocols for content identifiers. Further, it unifies disparate sources of metadata, as well as different metadata formats. In short, it provides cross-platform content identity and metadata services. A typical usage example to leverage these capabilities is as follows: a user wishes to find all metadata relating to a particular concert tour asso-



US 2007/0156726 A1

Jul. 5, 2007

13

ciated with a particular song; the user submits the user preference “concert tour” and “year=2005”) for the song; the reader extracts one or more content identifiers from the song, packages them with the user preferences, and sends a request to the routing system; the routing system establishes a federated content identity for the song knowing the ID providers from the reader and the content IDs; the routing system uses this content identity to find linked sources of metadata in its database (e.g., gets the URLs of these metadata sources); the routing system aggregates the metadata responses from these URLs from different domains; the routing system applies the user’s preferences and returns a metadata responses that relate to concert tour and the year 2005 to the user. The user need not concern herself with the details of identifying the content or gaining access to disparate metadata sources.

Public Interfaces for the Metadata System, Mash Ups for Metadata and Metadata Usage Statistics

[0148] The routing system establishes an interoperable system for associating content with metadata across disparate content identification and metadata systems. In so doing, it also provides a mechanism for monitoring consumption of content and metadata across disparate domains of content origin and consumption. The support for user generated metadata further enables new sources of metadata which users will likely value as much or more than the content itself. This is particularly true for a system that analyzes metadata usage and content usage and makes information available as additional metadata linked to the content because it facilitates the user’s ability to search for content that is of particular interest to him. One of the most compelling types of metadata is the metadata that identifies content by its popularity within certain groups (e.g., popularity as defined by tastemakers, affinity groups, niche genres for content, etc.).

[0149] In view of this important function of identifying compelling metadata, there is a commensurate value in an interface that efficiently unlocks this value by establishing programmatic access to the metadata and metadata statistics. Within a single system for metadata generation, programmatic interfaces are provided for that particular system. However, an embodiment of the routing system described in this document provides a programmatic interface across heterogeneous content identification technologies and metadata sources, and further, provides a programmatic interface to the metadata usage data that is collected across these heterogeneous metadata sources and linked back to individual content items through a variety of content identification technologies.

[0150] Examples of programmable interface technology that the routing system leverages include mash-ups built on Web 2.0, Programmable web, and semantic web programming constructs and Application Programming Interfaces (APIs). The implementer can use these technologies to build the metadata response aggregator, traffic monitor, and rule processor detailed above, as well as to build their own versions of these modules on top of the routing system. For example, the system can be embodied in a hierarchy of mash-ups, each providing additional functionality on top of the APIs of other mash-ups of metadata services provided by the system. The routing system, by providing a mechanism of linking metadata sources to content identifiers, creates a

Content Object Name Service (think of it as a Domain Name Service for content items). Further, the response aggregator, rule process or and traffic monitor build on top of this service, which as noted, can each be implemented as mash-ups of the routing system. When these metadata services are constructed in this form of interconnected mash-ups, the routing system provides a form of DNS for content metadata mash-ups.

Measuring Metadata Popularity and Tastemaker Identification

[0151] The tracking of content and metadata consumption in the system and subsequent publishing of this data to the user community enables user’s to more effectively identify metadata that matches their preferences. It also provides an opportunity for content owners to analyze these preferences and re-package content with popular user generated metadata. Content owners can also identify unique affinity groups and tailor new releases to the preferences of these groups. As noted above, the system tracks several forms of usage data. These types of usage data include monitoring traffic of content objects (e.g., tracking object distribution), traffic of metadata requests (e.g., number of requests for particular metadata, or metadata from a particular source), and user preferences and rules processing (e.g., user preferences derived from metadata analysis, such as the affinity group analysis). With the support for dynamic metadata (metadata generated over time as an object is consumed) and the means to measure popularity of metadata and its providers, the system identifies high value metadata which provides an opportunity for content object owners and advertisers to package this popular metadata and monetize it (sell it or sell advertising presented with its consumption by users).

[0152] The system also is able to capture a historical archive of metadata generation and consumption data, which further allows content and metadata distribution to be analyzed to identify transient metadata captured from points in time to be packaged and monetized in a similar fashion. For example, dynamic metadata can be packaged with the content objects that caused its generation and distributed. Examples in include content packaged by tastes of an affinity group of users, a period of time, a geographic market, etc.

CMDS Example Implementation Summary

[0153] The Content Metadata Directory Services (CMDS) provides a global trusted directory service that connects consumers of identified content to content-provider authorized and managed metadata databases and other digital resources.

[0154] It solves the problems created by the fact that digital distribution separates content from packaging, new 1-1 marketing opportunities are not being optimally utilized for content distribution, and digital distribution is moving forward with proprietary channels that make the value chain more complex rather than simpler. The CMDS system provides all existing value chain participants an environment to agree upon metadata usage and manage their proprietary metadata, as opposed to being another proprietary metadata repository. It also provides cross-sell/up-sell e-commerce opportunities. The CMDS system is interoperable with all content identifiers, PC and mobile devices, and enables usage reporting with vital marketing statistics.

[0155] The CMDS system standardizes three components (1) Registration Interface, (2) Resolution Interface and (3)



US 2007/0156726 A1

Jul. 5, 2007

14

Router Requirements. The interfaces are specified in terms of XML-based Web Services, an existing industry standard, for simplicity and interoperability. The router requirements guarantee that the system functions properly and maximizes value to vendors and users. These minimal specs create a system that is simple for vendors and users to interact with, while providing extremely flexible workflow and architecture. For example, CMDS can either (1) create unique content identifiers (CIDs) that can be embedded with any technology, such as digital watermarks (DWM) or signed headers, or (2) utilize CIDs created by pre-existing systems such as content fingerprints (a.k.a. robust hashes), Electronic Product Codes (EPC), IFPI's Grid, and URI. Furthermore,

users can learn more about artists, similar content and related items, and purchase content and related items.

**[0156]** The CMDS system provides users with valuable information and simple purchase options, and helps content owners and retailers increase ROI by expanding their knowledge of content usage and making it easier for consumers to buy content and related items.

#### CMDS Implementation Definitions

**[0157]** The terms used in the following sections are defined here, in alphabetical order.

Term	Description
Central Database	A global, master Database (where Database defined below).
Central Handler	A global, master Handler (where Handler is defined below).
Central Router	A global, master Router (where Router is defined below).
Content ID (CID)	A unique content identifier. It is different for different pieces of content, and can be unique for different copies of the same content. The combination of the CID with ID Provider ID and ID Version is globally unique. These additional fields allow ID Providers to embed and detect a smaller namespace (i.e. fewer bits) and the system to work with all pre-existing ID systems. However, these additional fields also require slightly more complexity in the Router.
Content Metadata	A system for Content Providers to register Content IDs and URLs, and Users to resolve Content IDs into URLs to obtain more information about the content where the Content Providers manage their proprietary metadata.
Directory Services (CMDS)	
Content Provider	The provider of the content, such as the content owner and retailer, or any company with rights to distribute content and/or content metadata.
Database	The second component of a Router which is used to store the Content Provider's links to their proprietary metadata, and store vendors' or users' contact information. Note: Often the term is used in the plural because it can refer to several linked databases.
Digital Watermark (DWM)	Data embedded within the content that does not degrade the content's quality to end-users, but is reliably detectable by enabled hardware or software.
EPC	Electronic Product Code as specified by EPCGlobal.
Fingerprint	A fingerprint identifies content from features of the content. It is also known as a robust hash or content based identification (and should not be confused with a forensic DWM).
Handler	The first component of a Router which handles the registration or resolution requests and responses. Note: Often the term is used in the plural because it can refer to several linked handlers.
ID Provider	The company that provides the technology to identify the content and interface with a Router.
ID Provider ID	The unique ID assigned to the ID Provider. An ID rather than name is used so that the combination of ID Provider ID, ID Version and CID is a globally unique number (when the CID is a number, such as when created by this system), and numbers are faster to lookup in a Database. (If the CID is pre-existing, such as URI, it may be formatted as text.)
ID Version	The ID version provides the version of the CID algorithm. It enables the ID Provider to re-use the same IDs in different version of their algorithms, such as for different content types, as well as to use various ID formats. The ID Version is different for each algorithm that uses overlapping CIDs for an ID Provider.
Mirror Database	An exact duplicate of the Central Database.
Mirror Handler	An exact duplicate of the Central Handler.
Mirror Router	An exact duplicate of the Central Router.
Resolution Request Message	The interface defining the message sent to a Router to request URLs (or other information).
Resolution	The interface defining the message sent from a Router in response to a resolution request.
Response Message	
Registration	The manager of the CMDS system. For a public system, as used in B2C environments, the Registration Authority is a trusted 3 <sup>rd</sup> party vendor. There is one public Registration Authority. For a private system, as used in B2B environments, the Registration Authority is usually the private system provider. There may be numerous private Registration Authorities.
Registration Authority	
Registration Request Message	The interface defining the message sent to a Router to request registration of CIDs, URLs, vendors or users.
Registration Response Message	The interface defining the message sent from a Router in response to a registration request.

US 2007/0156726 A1

Jul. 5, 2007

15

-continued

Term	Description
Request Code	A key part of the registration or resolution request message that describes the desired action for the message.
Router	The backend system that handles registration and resolution. It includes two main components, a Handler and Database. Note: Often the term is used in the plural because it can refer to several linked routers.
User	The end user of the system. For example, it may be a consumer linking to more information via their PC Multimedia jukebox, or a movie critic linking to current marketing materials via a closed, private system.

### CMDS Background

[0158] The Content Metadata Directory Services (CMDS) system is needed since content identification technology cannot provide useful information without accessing a back-end system that links the ID to relevant information (a.k.a. metadata). It is a router-based system, which is beneficial to a central metadata repository, so that content providers can manage their proprietary information and content can be routed to this information from any location.

[0159] CMDS provides a global trusted metadata directory service that connects consumers of identified content with authenticated "origin of source" databases and other content-provider authorized digital resources.

[0160] CMDS enables content owners to:

[0161] Leverage in-house digital asset management system

[0162] Gain from economies of scale

[0163] Act as the content authority over their own digital assets; and

[0164] Address digital distribution issues from a single, unified approach, rather than a fragmented approach.

[0165] Cross-sell/up-sell

[0166] CMDS also provides consumers metadata and e-commerce opportunities

[0167] Focused on routing

[0168] Content Providers manage their proprietary information

[0169] Enable eCommerce for all value chain participants

[0170] e.g. Both content owners and retailers can embed CIDs

[0171] Interoperability with all content identity provider technology

[0172] Compatible with both PC and mobile devices

[0173] Interoperability for multiple ID Providers whom license a common algorithm

[0174] Enable usage reporting and vital marketing statistics

### CMDS System Embodiment Overview

[0175] A content identification system has five main components: registration, embedding/calculating, reading, reso-

lution, and a router, as shown in FIG. 4. These components can be grouped into content identification and content routing (e.g., CMDS) categories. The content identification components include embedding content identification (or calculating it for fingerprinting) and reading content identification (a.k.a. detection). The content owner usually handles the embedding and the consumer product usually handles the reading.

[0176] The CMDS components include:

[0177] 1. Registration Interface

[0178] 2. Resolution Interface

[0179] 3. A Router

[0180] These three components are standardized in the companion specification such that any content identification technology can interoperate. The example implementation is also optimized for both technologies, such as DWM, where the fewer bits in the identifier the better, and for technologies, such as URI, which have non-integer namespaces. The example implementation has the flexibility to be optimized for a PC or mobile environment (or any future hybrid environment).

[0181] In FIG. 4, the content is labeled Protected and Identified Content since it is identified when distributed due to this process, and may also be protected by other means. Although the focus of this example implementation is using identification to enhance content, the identification may also be used for protecting the content and/or other technology may be used to protect the content. When the identification is used to both enhance and protect the content, there are mutual benefits, such as pirated content that doesn't have the identification does not provide the enhance features to the user. In other words, this implementation is not a replacement for content protection, but synergistic with it.

[0182] A Registration Authority handles these three content routing components. For a public system, as used in B2C environments, the Registration Authority is a trusted 3<sup>rd</sup> party vendor. There is one public Registration Authority. For a private system, as used in B2B environments, the Registration Authority is usually the private system provider (and usually the only ID Provider). There may be numerous private Registration Authorities, and each system most likely interact with another system in this B2B environment.

[0183] This system's design allows the Content Providers (including content owners, such as Record Labels, Movie Studios and Stock Photo Agencies, and retailers, such as Apple iTunes, Napster, WalMart, Hollywood Video, etc.) to

US 2007/0156726 A1

Jul. 5, 2007

16

interact directly with the user, such as a consumer, as shown in FIG. 5. This enables the Content Providers to manage their own proprietary data. In other words, the usage model is the user interacts with a router, which redirects the user to the Content Provider for metadata and e-commerce opportunities.

[0184] Thus, the system routes requests for metadata to Content Providers, such as content owners and retailers. In other words, the Router includes a Database that mainly contains CIDs and URLs that link to metadata and e-commerce opportunities, which are stored by the Content Provider, not at the Router.

[0185] The system has a distributed architecture, as shown in FIG. 7. At this time, only the duplicates of the Central Router (labeled Mirror Routers) are specified. These Routers can have separate Registration and Resolution Routers. There will most likely be more Resolution Routers since there will be more resolution requests. In the future, Cached Routers will interact with the Mirror Resolution Routers and only save recent resolution requests for efficient responses. Cached Routers are not applicable to registration requests since these requests occur neither often nor repetitively and are immediately forwarded to the Central Router. The example implementation requires the reader to periodically request the address of the Router that it should be utilizing for registration and resolution; thus, the system can dynamically change configuration. It is unnecessary for the Central Router and Mirror Routers to request addresses, as they are all run by the Registration Authority and, thus, know each other's addresses.

[0186] A Router consists of a Handler and Database. A Handler accepts the registration and resolution requests, quickly obtains the required information and sends the registration and resolution responses. It may be a single software thread running on the same machine as the database, such as for a local low-quantity implementation, or it may quickly hand-off requests by request code or ID Provider (as both fields are in the XML message header) to various request handlers that are linked to a local request Databases across multiple CPUs, as shown in FIG. 6.

#### Workflow Example

[0187] An exemplar workflow demonstrating content owners and retailer registering contact information and content, and consumers linking to metadata and purchasing content is shown in Table 1. The example assumes that the ID Provider and two Content Providers, a content owner and retailer, have received their respective identification and passwords from the Registration Authority. It also assumes that ID Providers have registered contact information (via UpdateIDProvider request code), and distributed embedder and reader software. The request codes are shown in italics between the arrows for messages sent to the router.

TABLE 1

Workflow Example		
Content Provider	Router	User
1. Content owner registers initial contact information →UpdateContentProvider→		

TABLE 1-continued

Workflow Example		
Content Provider	Router	User
2. Content owner notified of success ←		
3. Content owner creates a CID →CreateCID→		
4. Content owner receives CID ←		
5. Content owner registers Primary URL →RegURL→		
6. Content owner notified of success ←		
7. Content owner registers additional URLs →RegURLs→		
8. Content owner notified of success ←		
9. Retailer registers initial contact information →UpdateContentProvider→		
10. Retailer notified of success ←		
11. Retailer registers a pre-existing CID →RegCID→		
12. Retailer notified of success ←		
13. Retailer registers Primary URL →RegURL→		
14. Retailer notified of success ←		
		15. Consumer A listens to his PC Jukebox and clicks on the "more info" button based upon the URI in the audio content ←ResURLs←
		16. Router provides consumer with URLs from content owner and retailer, some of which create dynamic web pages →
		16(a). Consumer A learns more about the content and later purchases a CD by the same artist at the store.
		17. Consumer B decides to register to win a hybrid car ←RegUser←
		18. Consumer B is notified of success →
		19. Consumer B uses cell phone to learn more about a film via taking a picture of an ad and reading a DWM ←ResURLs←
		20. Router forwards URLs to consumer B's cell phone. Since the user is registered, the URLs includes lists of local theatres playing the movie and favorite local stores selling the DVD →
		20(a). The consumer buys the DVD from a local store!

#### Architecture Example

[0188] The architecture is flexible since the specification defines an extensible messaging interface and router design.

[0189] An example architecture which enables both content owners and retailers to use the system is shown in FIG.

US 2007/0156726 A1

Jul. 5, 2007

17

8. Many other architectures or expansion of this architecture are possible, but this architecture is expected to be the most used architecture. This architecture enables the workflow discussed in the previous section, as well as many other workflows. The architecture includes:

- [0190] Content owner environment with embedding/calculating software that has a registration interface to embed identification in the content (or store its fingerprint)
- [0191] Retailer environment with embedding/calculating software that has a registration interface to embed or add identification to the content (or store its fingerprint)
- [0192] A Router consisting of a Central Handler, Central Database, Mirror Handlers and Mirror Databases
- [0193] Protected and identified content in cyberspace
- [0194] User Environment with reader software that has a resolution interface to link the user to more information, as well as a registration interface to register the user, if the user opts-in to send secondary personal information

#### Usage Scenario Examples

[0195] These usage scenarios help demonstrate the flexibility and capabilities of the specification. They also discuss what technically happens in the background (with request codes and other XML tags in parenthesis)—all in italics since they are technical details.

#### Consumer linked to Metadata and eCommerce

[0196] This usage scenario includes most aspects of registrations for ID Providers, Content Providers, e.g. content owners and retailers, and Users, as well as Users, e.g. consumers, linking to metadata and eCommerce. It demonstrates several key aspects of the system, and is most similar to the workflow example in Table 1.

#### ID Provider Registration

[0197] The first step is registration of at least one ID provider. The ID provider receives an ID and password from the Registration Authority via a secured process, such as a mailed letter or telephone call and company and contact person security check. Next, the ID provider securely registers their contact information with the registration authority, either through the Registration Authority's software (via UpdateIDProvider) or through an interactive secure web page. The ID Provider is notified of success or any errors.

In the background, the ID Provider embedder and detector software contains the ID Provider ID, and submits it, along with an ID Version of the embedding or detection algorithm, with every request.

#### Content Owner Registers as Content Provider

[0198] The second step is registration of a content owner. The content owner receives a Content Provider name and password from the Registration Authority via a secured process, such as a mailed letter or telephone call and company and contact person security check. Then, the content owner uses their embedder or related software received from the ID Provider to securely register contact information (via UpdateContentProvider). The embedder

software verifies the response, and notifies the content owner of success or any errors.

[0199] Alternatively, the content provider could have used the Registration Authorities secure web site to interactively register their contact information.

#### Content Owner Registers New Content

[0200] Once the Content Owner has registered, they securely register their content. First, they use the embedder or related software to obtain a unique Content ID (via CreateCID). Afterwards, the Content Provider registers two primary URLs, one for PCs and one for mobile devices (via URLType XML tag with Full or WAP data entry), as well as four additional URLs, two for PC and two for mobile, (via six calls with RegURL request code). For example, for both PCs and mobile devices, the primary URL provide static information about the movie Fantasia from Disney, one additional URL provides information about songs in Fantasia, and the other additional URL provides items for sale regarding Fantasia from Amazon.com, using a dynamic web page via a URL including search terms based upon the CID. The software application verifies the response for success and let the content owner know about the success (or any errors).

[0201] In one alternative, the content owner could enter multiple URLs in the software's GUI with checkboxes for fully functional or WAP. In the background of this configuration, the software uses the request grouping method to obtain the CID (via CreateCID), register the primary URLs (via two calls with RegURL), and additional URLs (via two calls with RegURLs). The Routers handle the timing such that the CID is registered before the URLs.

[0202] In another alternative, the content owner can use the Registration Authorities secure web site, and interactively register the URLs, one-by-one.

#### Retailer Registers as Content Provider and Registers Content

[0203] A retailer, assuming they have worked out rights with the content owner, can also securely register as a Content Provider with the Registration Authority, obtain a Content Provider name and password, and register contact information (via UpdateContentProvider). For example, the retailer could be Apple iTunes store. Then, when the retailer prepares to sell content, they request a CID (via CreateCID) and add two URLs, one primary URL to sell songs by the same artist (via RegURL), and one additional URL to sell similar songs (via RegURLs).

#### Anonymous User Linking

[0204] When a user receives a copy of the content with the registered CIDs, the user can request and receive more information from their multimedia player. Preferably, the multimedia player includes a reader plug-in that always scans for CIDs, checks that URLs are registered for this CID, and displays a symbol or "more info by <company name>" icon to let the user know that the content is enhanced. This scanning process also makes the response time immediate since CID detection has already been performed.

[0205] When the user selects to receive more information, the user receives five web links with brief descriptions.



US 2007/0156726 A1

Jul. 5, 2007

18

There are three links from the content owner: one with the film history, one with related song information, one with items for sale regarding Fantasia from Amazon.com. There are two links from the retailer whom sold them the movie: one where they can buy songs by the same artist, and one where they can purchase similar content. The user can click on the links to visit the displayed web sites.

[0206] In the background, the reader software requests the URLs (via ResURLs) with a group containing two requests, one for the content owner's CID and one for the retailer's CID. As the last step, the reader software parsed the returned URLs and displayed them to the user.

[0207] If the CIDs are embedded by two different ID Provider technologies, the user will see two different "more info" buttons, e.g. one "more info from Disney" and one "more info from Apple iTunes Store" and the depending upon the user's selection, the corresponding request is sent and response is displayed.

#### Mobile User

[0208] When a cell phone user identifies content, he/she links to the WAP or WMI version of the registered URLs, assuming the Content Provider registered such URL types (via XML Tag URL Type with WAP or WMI data during registration or resolution). Information, formatted for their small screen, is displayed. In addition, preferably, the user can select the display of only a primary link (for content owner or retailer) so that the Content Provider's information is displayed immediately after selecting to receive more information (without the user having to select again from a list of URLs and descriptions since they may be driving).

[0209] For example, when the cell phone user hears a song that they like on the radio, they can identify the song (via a DWM or fingerprint) and link to more information. Similarly, the mobile user can identify songs they are listening to on their PDA phone and link to more information.

#### User Registration

[0210] Another user decides to register personal information to obtain a chance to win a new hybrid car. They register from the reader plug-in for their multimedia player (via RegUser). The software application verifies the response for success and lets the user know about the success (or any errors). The user can update their information at any time (via UpdateUser), or decide to opt-out for one specific link or all future links.

#### Registered User Linking

[0211] When this registered user requests more information about the video from the reader plug-in software (via ResURLs), the same five links as displayed for the anonymous user are displayed, as well as links to additional theatres in the user's zip code that are playing Fantasia or similar movies, as well as stores in the user's zip code that carry the related merchandise. In addition, if the Content Provider (e.g. content owner or retailer) maintain information for each username, they can suggest other items based upon the user's previous purchases, or produce a "instant chat" icon if this is a preferred customer who has purchases several things in the past (equivalent to nicely dressed shopper or frequent shopper card holders getting better service in the physical store). Someday this user believes

they will be one-click away from purchasing any of the listed items, when all stores share the same secure user information.

#### Content Provider and ID Provider Reports and Billing

[0212] At the end of the month (or any other time), the Content Provider (e.g. content owner or retailer) logs onto the Registration Authorities web page and securely views usage data for each of their registered CIDs. They see that their new movie is hot in the northeast part of the USA and Canada, especially by 8-to-10 year olds around 7 PM EST. They also see that its usage has been starting to decline, and represents 20% of their current resolution usage for this age group, indicating that it is time to release a new movie with this target audience. Finally, they can see their costs and billing options, such as automatic withdrawal at fixed amounts or monthly billing.

[0213] The ID Provider can also log onto the same web site and view aggregate usage data for all of the CIDs registered using their technology. However, they cannot view stats by individual CIDs since that is owned by the Content Provider.

[0214] In the background, each Router is saving log files for each hit, and, preferably, daily processing the log files to aggregate usage data to provide real-time interaction with these stats, as well as automatic capabilities to email reports to the Content Provider and ID Provider.

#### Fingerprints/Robust Hashes Link Content to URLs

[0215] When using fingerprints to identify content, the fingerprint is calculated from the content and entered into a database. Then, when trying to identify content, the fingerprint is calculated, and matched to the closest fingerprint in the database that also falls within a certain threshold of certainty. The fingerprint is usually a collection of sub-fingerprints, especially when content is streaming. As such, the fingerprint is not a unique content ID, but the fingerprint database links the fingerprint (i.e. collection of sub-fingerprints) to a unique content ID.

#### Content Provider Registration

[0216] The Content Provider can either (i) utilize the fingerprint provider's proprietary content ID system and register pre-existing CIDs with a Router (via RegCID), or (ii) request a unique CID (via CreateCID) and use it in the fingerprint database. Then, the Content Provider can register URLs to link to the CID (via RegURL and RegURLs).

[0217] In case (i), CID uniqueness is guaranteed since the fingerprint database guarantees the CID is unique in that database, and the combination of the CID with ID Provider ID and ID Version guarantee global uniqueness. In case (ii), the CID is guaranteed to be unique by the Central Router, and the same combination is globally unique. The ID Version is uniquely defined for each fingerprint algorithm and related database for each ID Provider.

#### Linking Users to Metadata with Fingerprints

[0218] The user selects to receive more information as with any CID and receives URLs and short descriptions (via ResURLs). The user then clicks on the URL with the information that he/she wants to receive.

US 2007/0156726 A1

Jul. 5, 2007

19

[0219] In the background, when a user requests to receive more information for content identified with fingerprint, the fingerprint reader calculates the fingerprint, sends it to the fingerprint server (either running on the fingerprint provider's server or on the Central Router), the fingerprint server determines the CID from the fingerprint database, returns the CID to the reader, and the reader uses the CID to request the URLs (via ResURLs) from the Router. The Router, having access to the IDProviderID, ID Version and CID data in the resolution request message, uses this globally unique combination to lookup the correct URLs. The proper ID Version is known by the reader since it used that version of the algorithm to determine the CID.

Linking via Other ID Systems, such as EPC

[0220] In some cases, a different Registration Authority will have previously provided a unique identity, possibly for other purposes, and it is optimal to be able to use this unique content ID in this system. For example, the Electronic Product Code (EPC) provides unique IDs with radiofrequency identifiers (RFIDs). Similarly, IFPI's GRid provides unique content IDs. Or, 4C provides unique IDs with Verance's DWM. Furthermore, an ID Provider could generate their own unique ID, such as using the "album:artist:song" combination from ID3 tags.

[0221] An example 96 bit EPC format, also known as general identifier (GID-96) is:

[0222] Header: 8 bits

[0223] Manufacturer: 28 bits or 268 million unique IDs

[0224] Product (SKU): 24 bits or 16 million unique IDs

[0225] Serial Number: 36 bits or 68 billion unique IDs

[0226] As the goal of CMDS is to link users' to information managed by content owners and identified by multiple ID technology providers, whereas the goal of EPCNet involves tracking items through the distribution process for each participant, the backend structures are different. Thus, these two networks can work synergistically, such as with the EPCNet IS having an interface to a CMDS Router. An example of using EPC, such as a film poster ad containing an RFID with EPC GID-96 format, is described below.

Register URLs for Pre-existing CID

[0227] The first step is that the Content Provider registers the pre-existing CID (via RegCID). Then, the Content Provider registers a primary Full URL (via RegURL), and, if desired, they can register additional URLs (via RegURL or RegURLs) or URLs for mobile formats.

[0228] When using this specification to create CIDs, the CID namespace is guaranteed to be unique for each ID Provider and their current ID Version (e.g. algorithm version), and the CID format is decimal. However, since the pre-identified content may have CIDs that overlap other CIDs used by the same ID Provider in this system, the ID Version (via IDVersion XML tag) is used to determine the proper CID namespace. For example, a CID representing a GID-96 EPC code has the hex format of HH.MMMMMMMPPPPPP.SSSSSSSS where H is header (8 bits), M is Manufacture (28 bits or 268 million unique IDs), P is Product SKU (24 bits or 16 million unique IDs) and S is Serial Number (36 bits or 68 billion unique IDs). Thus, the ID Provider has an ID Version for the EPC

GID-96format, and different ID Versions for other EPC formats or other CID formats (e.g. DWM). When the embedder registers an EPC GID-96 format CID, it sends the corresponding ID Version.

Link to URL with Pre-Existing CID

[0229] The reader software detects the pre-existing CID, sends it to the router (via ResURL), receives URLs and displays them to the user.

[0230] In the background, the router, having access to the IDProviderID, ID Version and CID data, uses this globally unique combination to lookup the correct URLs. The proper ID Version is known by the reader since it needs to use that version of the algorithm to detect the CID.

Interoperable ID Technology from Multiple ID Providers

[0231] When one company licenses embedder and detector technology (i.e. OEM) to multiple ID Providers, there arises the user-desired solution where one ID Provider's reader can utilize a CID embedded with another ID Provider's embedder. However, the ID Provider desires "credit" for the embedding process; thus, this content routing system tracks the embedding ID Provider.

ID Provider ID Embedded in Content

[0232] Each ID Provider's embedder embeds their ID Provider ID along with the CID that is requested and embedded by the Content Provider (via CreateCID).

Embedding ID Provider ID Included with URL Requests

[0233] When the reader software reads a CID, it also reads the embedding ID Provider ID. If the embedding ID Provider ID is different than that stored in the reader, it is included in the URL resolution request message (via the EmbIDProviderID XML field). The Router logs the embedding ID Provider's ID for any further action, such as being properly compensated for the embedding process.

CMDS Section Conclusion

[0234] In conclusion, the companion specification provides valuable information to consumers, and enables a distributed routing architecture such that Content Providers manage their proprietary data and interaction with the consumer. Content Providers can include content owners, such as Record Labels, Movie Studios, stock photo agencies, etc., and retailers, such as Apple iTunes, Walmart (clicks-and-mortar), catalog companies, advertisers, Netflix, etc.

[0235] Furthermore, the specification enables Content Providers to use any identity provider technology to link users to the proper information. In addition, when multiple ID Providers are sharing a content identification OEM algorithm, every router can seamlessly interoperate and track usage for proper payments. Importantly, the system enables multiple CIDs to exist in content such that the complete value chain (e.g. content owners and retailers) can participate, or create business rules for participation.

[0236] Furthermore, the specification is optimized for all identification technologies and existing systems, including digital watermarks (DWM), fingerprints (a.k.a. robust hashes), Electronic Product Codes (EPC), IFPI's Grid, and URI. Finally, the specification includes logging, not only for security tracing, but also for usage report that help ID Providers understand router demands and Content Providers

US 2007/0156726 A1

Jul. 5, 2007

20

understand content usage. The specification can easily be expanded to handle usage cases around seeding multiple ID systems and providing “buy now” functionality for non-legitimate content.

[0237] In the end, the specification should help content owners and retailers (thus distributors) increase sales (fROI) by expanding the knowledge of content usage and making it easier for consumers to buy content and related items.

[0238] The following example implementation uses industry standard XML-based Web Service to provide an open interface to the Routers.

#### Overview of Example CMDS Implementation

[0239] The main components of the specification of this example CMDS implementation include:

[0240] Registration Interface

[0241] Registration Request Messages

[0242] Registration Response Messages

[0243] Resolution Interface

[0244] Resolution Request Messages

[0245] Resolution Response Messages

[0246] Router Requirements

[0247] The registration and resolution request messages use XML and include:

[0248] Header

[0249] Body

[0250] Primary Information

[0251] Secondary Information

[0252] The response messages are simpler. The registration response message is simple, requiring no header or body sub-sections, and the resolution response message only includes primary and secondary information sub-sections.

#### ID Providers, Content Providers, and Users

[0253] To understand this example implementation, it is helpful to understand the relationship between ID Providers, Content Providers and Users.

ID Providers: Companies that provide the technology for content identification and to interface with a Router.

[0254] Content Providers: Providers of the content, such as the content owner and retailer, or any company with rights to distribute content and/or content metadata. For example, content owners can include Record Labels, Movie Studios, and Stock Photo Agencies. Retailers can include Apple iTunes, Wal-Mart (click-and-mortar), advertisers, catalog companies, and Napster.

Users: The end users of the system. For example, it may be a consumer linking to more information via their PC Multimedia jukebox, or a movie critic linking to current marketing materials via a closed, private system.

CID, ID Version, ID Provider ID and Globally Uniqueness

[0255] Furthermore, this example implementation is based upon a unique content identifier, labeled a CID. It is critical that CIDs are unique for each ID Provider ID and ID

Version. CIDs can overlap for different ID Provider IDs or different ID Versions. ID Provider IDs are different for each ID Provider, and an ID Provider could register multiple ID Provider IDs, if determined as needed by the Registration Authority.

[0256] The ID Version is the version of the technology/algorithm that embeds and reads the CID. The ID Version is always known by the embedder and reader since they determine which algorithm to embed or read the CID. The ID Provider can have CIDs that overlap when they have different ID Versions, and uses different ID Version when CIDs can overlap.

[0257] For example, a video may have the same CID as a song, or one song embedded by ID Provider 1 may have the same CID as a different song embedded by ID Provider 2 (or by ID Provider 1 using a new ID Version).

[0258] As such, CIDs are not globally unique unless combined with these other variables. Thus, the combination created by appending IDProviderID, IDVersion and CID together (e.g. appending <IDProviderID><IDVersion><CID>) is a globally unique ID, usually a number.

[0259] Alternatively, the globally unique combination can be represented as a URI, formatted as “CMDS://<IDProviderID>.<IDVersion>.<CID>”.

[0260] The advantages to dividing the namespace in this format are twofold: (1) fewer bits are needed to represent CIDs since CIDs can be re-used in new content or new algorithm versions, and (2) the system is easier to integrate with pre-existing identity systems such as EPC, fingerprinting and 4C/Verance DWM since the CIDs can overlap between these systems but be differentiated with the ID Version.

[0261] Finally, for many identification technologies, the CID is an integer, and it is most efficient for transport and backend/router processing to use an integer field. However, for other identification technologies a text field is required, and the router can process the text into a unique content identifier. As such, the CID format is determined by ID Version data, and properly handled by a Router. As an aside, two XML schemas will be present for these requests, one for an integer CID and one for a text CID. If possible, it is optimal to use the integer format. Along these lines, databases can be indexed by all three identifiers (IDProviderID, IDVersion and CID) as separate variables or databases, or one database can use this triplet ID as a globally unique index when the CID is integer format.

#### Registration Interface Specification

[0262] The registration process enables the content provider to obtain unique CIDs and link the CID to URLs. It also enables the ID Provider, Content Provider, and User to register contact information.

[0263] The registration works from both (i) an interactive human-readable web interface and (ii) a web services interface that interacts directly with registration software, such as a CID embedder (or multimedia plug-in for user registration) that runs on the vendor’s or user’s computer.



US 2007/0156726 A1

Jul. 5, 2007

21

## Secure Authenticated Channels

[0264] The client web and software interface uses a secure authenticated channel with the Router. This protection is required so that no one but the proper vendor can change the registration data.

## Content Provider, ID Provider and User Registration

[0265] Content Providers and ID Providers initially register with the Registration Authority via a process that verifies them and the contact person to be a trusted provider. During this process, the Content Provider is assigned a unique vendor-name and password, and the ID Provider is assigned a unique numeric ID and password.

[0266] ID Providers are assigned a unique number rather than name for the following reasons: it is quicker for a Router to lookup a number, the ID Provider's software remembers the number (so the human readability is not important for remembering the unique identification), and the globally unique combination of IDProviderID, IDVersion, and CID is a number.

[0267] Once they receive their vendor info and password, Content Providers and ID Providers can update their contact information via their embedder software or directly with Registration Authorities interactive web site.

[0268] Users register via the reader software, which registers a username and password, and should provide the user the option to store the password for future usage. The reader software keeps the username and submits it with each resolution request, as well as offers the user the option to block sending the username (for one request or all future requests). In other words, this particular system is an opt-in system, with capabilities to opt-out.

## Registration Request Message

[0269] The Registration Request Message is the interface that defines the message sent to a Router for data registration. The Registration Request Message interface includes an XML Header and Body. Examples are shown below, and the format is described below.

## Registration Request Root Elements, Sub-Elements and Grouping

[0270] Registration request messages choose one root elements from several options, include:

[0271] <RegistrationCIDRequest> for CreateCID and RegCID request codes;

[0272] <RegistrationURLRequest> for RegURL, DelURL, RegURLs, and DelURLs request codes; and <RegistrationContactInfoRequest> for UpdateContentProvider, UpdateIDProvider, RegUser and UpdateUser request codes. (More root elements may be added when the handling of integer and text CIDs is finalized, e.g. RegistrationhntCIDRequest and RegistrationTextCIDRequest.) There are sub-elements, including: <Header> for message headers, <Body> for message body, and <PrimaryInfo> and <SecondaryInfo> when needed. Grouping of registration requests is allowed (and it is expected to be specified when the transmission method, e.g. SOAP, is finalized, as discussed below).

## Registration Request Message XML Header

[0273] The XML header is:

XML Tag	Format	Description	Status
Version	Integer	The version of this request message. The current version is 1.	Required
IDProviderID	Integer	Identity technology provider number as stored in the product (e.g. software) interacting with a Router. IDs 0-511 are reserved (as described below).	Required
RequestCode	Text	Requests a Router to take a specified action.	Required

[0274] The allowable Request Codes for registration include:

Request Code	Description
CreateCID	Create a unique content ID
RegCID	Register pre-existing CID from other unique identification standard
RegURL	Register one URL
DelURL	Delete one URL
RegURLs	Register several URLs
DelURLs	Delete several URLs
UpdateContentProvider	Update Content Provider contact information
UpdateIDProvider	Update ID Provider contact information
RegUser	Register new username, password and contact information
UpdateUser	Update user contact information

[0275] This header data can help the router quickly distribute the message to various request handlers for the detailed actions to be handled.

[0276] The request codes RegURLs and DelURLs register or delete multiple URLs in one message, which requires less interaction from the registration software but requires parsing by the Router.

[0277] The ID Provider IDs' 0-to-511 are reserved, such as for the case when the ID Provider embeds their ID within the content since there are several interoperable readers.

## Registration Request Message XML Body

[0278] For each Request Code, the Resolution Message XML Body includes the following information.

## Request Codes: CreateCID and RegCID

[0279] The XML body has primary information, which includes CID related variables, for the registration request, and secondary information which includes optional descriptive data for the CID.

The primary request information is:

[0280] The CID Expiration field should be used with great caution since content may exist for a long time. It is most relevant to temporary content, like news papers or catalogs.

US 2007/0156726 A1

Jul. 5, 2007

22

[0281] The secondary request information is:

XMLTag	Format	Description	Status
ContentProviderName	Text	Content Provider name.	Required
Password	Text	Content Provider password.	Required
IDVersion	Integer	The version of the CID algorithm.	Required
CID	Integer or Text	The unique content ID formatted as integer (preferable) or text, as determined by the ID Version.	Required for RegCID
CIDExpiration	Date	Expiration date for CID so it can be reused. Use with caution (as described below)	Optional
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

XMLTag	Format	Description	Status
Title	Text	Title of the content.	Optional
Copyright	Text	Copyright dates.	Optional
AdultFlag	Boolean	TRUE for adult content, and FALSE for all other content	Optional
ContentType	Text	Content type choices include: audio-music, audio-speech, video videotrack, video-audiotrack, image, and text.	Optional
ArtistName	Text	Artist's full name	Optional
ArtistEmail	Text	Artist's email address	Optional
ArtistPhone	Text	Artist's phone number	Optional
ArtistURL	Text	ID Provider, Content Provider or User Artist web page.	Optional

Request Codes: RegURL, DelURL, RegURLs and DelURLs

[0282] The XML body information is:

XML Tag	Format	Description	Status
ContentProviderName	Text	Content provider name.	Required
Password	Text	Content provider password.	Required
IDVersion	Integer	The ID version provides the version of the CID algorithm.	Required
CID	Integer or Text	The unique content ID formatted as integer (preferable) or text, according to the IDVersion.	Required
Primary	Boolean	TRUE for Primary URL, and FALSE for Additional URLs.	Required
URLType	Text	The URL Type, such as for a WAP enabled cell phone or fully functional PC (more details below).	Required
URL	Text	The URL that links the content to more information.	Required
URLVariables	Text	A list of XML tags, separated by a colon ":", that define the XML fields which are appended at the end of the URL or the term ALL for all of the request tags (more details below).	Optional
URLActivation	Date	Date to activate the URL. If no date is set, the URL is activated, and remains active unless it has expired.	Optional
URLExpiration	Date	Date to de-activate URL. If no date is present, the URL has no	Optional

US 2007/0156726 A1

Jul. 5, 2007

23

-continued

XML Tag	Format	Description	Status
Desc	Text	expiration, and is active, unless it has not yet been activated. Brief description of URL, less than 128 characters with spaces	Required, except for DelURL and DelURLs
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

[0283] The allowable URL Types (for all request codes that include this field):

URL Type Code	Description
Full	Fully functional web pages
WAP	Wireless Application Protocol (WAP) web page
WMI	W3C Mobile Web Initiative (MWI)

[0284] The URL Variables enables the URL to deep link into a database.

[0285] For RegURLs, a list of Primary, URLType, URL, URLVariables, URLActivation, URLExpiration, and Desc data elements are separate by semicolons “;”.

[0286] The fields has a entry for each URL (a space for no data is okay), or not be used at all

[0287] For DelURL and DelURLs, the data elements in URLVariables, URLActivation, URLExpiration, and Desc are optional, and are ignored if included.

[0288] For DelURLs, a list or Primary, URLType and URL data elements are separated by semicolons “;”.

[0289] The other fields, including IDVersion and CID, remains constant

URLs are categorized into Primary and Additional URLs, where there can be one Primary URL for each URL Type and as many additional URLs as desired. This categorization allows immediate redirection for the user, as well as choice of all associated URLs (i.e. additional actions) for the user. In other words, Primary URLs enable the system to automatically display the primary web site for the user, thus not requiring the user to click on the desired URL after selecting to receive more information. There is a balance, since while reducing the number of required clicks, Primary URLs also reducing the choices and retail opportunities.

Request Codes: UpdateContentProvider, UpdateIDProvider, RegUser and UpdateUser

[0290] The XML body information is:

XML Tag	Format	Description	Status
RegName	Text	Content Provider Name, Username, or ID Provider ID. The ID Provider ID is converted to an integer inside the router's handler.	Required
Password	Text	ID Provider, Content Provider or User password	Required
BizName	Text	Company's full name	Required, except for RegUser and UpdateUser
BizEmail	Text	Company's email address	Optional
BizAddress1	Text	Company's street address, first line	Required, except for RegUser and UpdateUser
BizAddress2	Text	Company's street address, second line	Optional
BizCity	Text	Company's city	Required, except with RegUser and UpdateUser
BizState	Text	Company's state	Required, except with RegUser and UpdateUser
BizZip	Text	Company's zip	Required, except with RegUser and UpdateUser
BizCountry	Text	Company's country	Required, except with RegUser and UpdateUser
BizPhone	Text	Company's phone number	Optional
BizURL	Text	ID Provider, Content Provider or User company web page.	Optional

US 2007/0156726 A1

Jul. 5, 2007

24

-continued

XML Tag	Format	Description	Status
BizLogoURL	Text	ID Provider or Content Provider company logo.	Optional
BizTemplateURL	Text	ID Provider or Content Provider company template.	Optional
NameTitle	Text	Contact's name title (e.g. Mr., Ms., etc.)	Optional
NameFirst	Text	Contact's first name	Required
NameMiddle	Text	Contact's middle name or initial	Optional
NameLast	Text	Contact's last name	Required
NameSuffix	Text	Contact's name suffix (e.g. JR., III, etc.)	Optional
Email	Text	Contact's email address	Required
Address1	Text	Contact's street address, first line	Required, except for RegUser and UpdateUser
Address2	Text	Contact's street address, second line	Optional
City	Text	Contact's city	Required, except for RegUser and UpdateUser
State	Text	Contact's state	Required, except for RegUser and UpdateUser
Zip	Text	Contact's zip	Required
Country	Text	Contact's country	Required
Phone	Text	Contact's phone number	Required, except for RegUser and UpdateUser
Cell	Text	Contact's cell phone number	Optional
Fax	Text	Contact's fax number	Optional
IM	Text	Contact's instant message address	Optional
Language	Text	Contact's preferred spoken language.	Optional
Sex	Text	Contact's Sex (M or Male, or F or Female)	Optional
Age	Integer	Contact's Age	Optional
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

[0291] List of countries from Windows XP "Region and Language Options" in Control Panel

[0292] List of languages from Windows NP "Region and Language Options" in Control Panel

#### Registration Response Message

[0293] The Registration Response Message is the interface that defines the message sent from a Router in response to data registration. The Registration Response interface includes a success code, URL and brief description of the URL, or error code and associated URL and description, as

well as private data. Examples are shown below, and the format is described below.

#### Registration Response Root Element and Grouping

[0294] All registration response messages have one root element of <RegistrationResponse> (and </RegistrationResponse>), and no sub-elements. Grouping of registration requests is allowed (and it is expected to be specified when the transmission method, e.g. SOAP, is finalized).

#### Registration Response XML Message

[0295] The registration response message XML format for all requests is:

XML Tag	Format	Description	Status
Version	Integer	The version of this request message. The current version is 1.	Required
RtnCode	Integer	Return code where "0" means success and any positive number as an error code. For this version, the only valid error code is "1".	Required
URL	Text	URL, or list of URLs separated by semicolons ";"	Optional
Desc	Text	Brief description of URL or error, or list of descriptions of URLs, with each description or error text having 128 characters or less including spaces, and separated by semicolons ";"	Required for errors
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

#### Registration Response Message Data

[0296] For the request for a unique ID (CreateCID): the message returns a "0" for success in the RtnCode field and the CID in the Desc field—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

[0297] For the request for registering CIDs from other system ID (RegCID): the message returns a "0" for success in the RtnCode field—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

[0298] For the request to register or delete a URL (RegURL, DelURL, RegURLs, and DelURLs): the message returns a "0" for success in the RtnCode field—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

For the request for content provider, ID provider or user registration

[0299] (UpdateContentProvider, UpdateIDProvider, RegUser, and UpdateUser): the message returns a "0" for success in the RtnCode field and RegName (e.g. ContentProviderName, IDProviderID, or UserName) in the Desc field for optional verification—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

#### Content Provider Display

[0300] The Content Provider is notified of the success or error. In the case of error, the error text and URL shall be displayed.

US 2007/0156726 A1

Jul. 5, 2007

25

## Resolution Interface Specification

[0301] The resolution architecture connects the readers to a Router such that content identification can be used to provide users with interesting related data and purchasing opportunities.

[0302] The resolution interface employs a web services interface.

## Secure Authenticated Channel for Request Address

[0303] The web services interface uses a secure authenticated channel with the Router for ResRegAddress and ResResAddress. This protection is required to eliminate spoofing of the Router.

## Resolution Request Message

[0304] The Resolution Request Message is the interface that defines the message sent to a Router for data lookup. The Resolution Request Message interface includes an XML Header and Body. Examples are shown and the format is described below.

## Resolution Request Root Elements, Sub-Elements and Grouping

[0305] Resolution request messages choose one of two root elements: <ResolutionURLRequest> for ResURL and ResURLs request codes; and <ResolutionAddressRequest> for ResRegAddress and ResResAddress request codes. (More root elements may be added when the handling of integer and text CIDs is finalized, e.g. ResolutionURLIntRequest and ResolutionURLtextRequest.) There are sub-elements, including: <Header> for message headers, <Body> for message body, and <PrimaryInfo> and <SecondaryInfo> when needed. Grouping of registration requests is allowed (and it is expected to be specified when the transmission method, e.g. SOAP, is finalized). Grouping is useful when the ID Provider reader can read multiple CIDs embedded in the content (e.g. one CID from a content owner and one CID from a retailer).

## Resolution Request Message XML Header

[0306] The XML header is the same as for a registration request. This header data can help the router quickly distribute the message to various request handlers for the detailed actions to be handled.

[0307] The allowable Request Codes for resolution include:

Request Code	Description
ResURL	Resolve the primary URL
ResURLs	Resolve the primary and additional URLs
Res2ndInfo	Resolve the secondary information
ResRegAddress	Resolve the address of the Registration Router for registration requests
ResResAddress	Resolve the address of the Resolution Router for resolution requests

## Resolution Request Message XML Body

[0308] For each Request Code, the Resolution Message XML Body includes the following information.

Request Code: ResURL, ResURLs and Res2ndInfo

[0309] For these request codes, the message includes primary and secondary information. The Primary Information portion contains the data required to properly service the request. The Secondary Information contains user-specific data and is intended for user specific links and aggregate usage monitoring and reporting.

[0310] Privacy issues should be considered when sending the secondary information. Preferably, the user permits the secondary information to be sent.

[0311] The primary request information includes:

XML Tag	Format	Description	Status
Timestamp	Time	Time stamp of triggering event. Used to a id in management of "stale" requests that have timed out. Format is Coordinated Universal Time (e.g. 2005-04-14T13:20:00Z)	Required
EmbIDProviderID	Integer	The ID Provider ID that embedded the CID, if different than that of the reader software ID Provider ID	Required if different than IDProviderID, otherwise blank
IDVersion	Integer	The ID version provides the version of the CID algorithm.	Required
CID	Integer or Text	The unique content ID formatted as integer (preferable) or text, as defined by the IDVersion.	Required
URLType	Text	The URL Type, such as for a WAP enabled cell phone or fully functional PC.	Required, except for Res2ndInfo
Response2ndInfo	Boolean	TRUE if secondary response information is requested, and FALSE if not. (It is not signaling that secondary request information is included in this request).	Required, ignored for Res2ndInfo
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

[0312] Private data could, for example, include part of an image or audio so the server can detect the embedded CID. The binary data format should be known by the ID Provider handler, as well as preferably included in the header of the binary data.

US 2007/0156726 A1

Jul. 5, 2007

26

[0313] The secondary request information includes:

XML Tag	Format	Description	Status
ReaderID	Integer	A unique ID that identifies the reader application per purchase or installation. (More details below)	Optional
Transaction ID	Integer	A unique ID that identifies the transaction for the reader. (More details below)	Optional
OS	Text	Operating system, including Windows, WinCE, Mac, Symbian, J2ME, BREW, PALM OS, Microsoft Smartphone/Pocket PC	Optional
OSVersion	Text	Request OS for version from OS and send in same format	Optional
GPS	Text	GPS coordinates of requesting device, if available	Optional
Username	Text	Identification of the user and links to data that the user has registered	Optional
sLanguage	Text	User's preferred spoken language.	Optional
sSex	Text	User's Sex (M or Male, or F or Female)	Optional
sAge	Integer	User's Age	Optional
sZip	Text	User's postal code	Optional
sCountry	Text	User's Country.	Optional

[0314] The Reader ID eliminates the statelessness of a Web Services request and enables useful usage tracking. It is optional due to privacy concerns and shall be able to be turned off by the user.

[0315] It can be calculated from the reader machine or pre-assigned via purchase/activation codes, where the latter is preferable for mobile devices that may not be able to generate GUIDs.

[0316] The Transaction ID provides further tracking upon the reader ID, and can also be turned off by the user.

[0317] It can be as simple as starting with 1 and counting each transaction by that reader.

[0318] Note that "s" in front of variables means that it is the secondary information, and allows anonymous data aggregation without a user registering.

[0319] If the Username is included, none of these "s" variables are needed since they have been registered by the user and are stored in a Router

[0320] sLanguage uses the list of languages from Windows XP "Region and Language Options" in Control Panel

[0321] sCountry uses list of countries from Windows XP "Region and Language Options" in Control Panel

[0322] As such, the following actions are enabled:

[0323] Aggregate usage monitoring and reporting

[0324] Secondary data about the user to be used for detailed usage monitoring and user specific resolution responses

Request Code: ResRegAddress and ResResAddress

[0325] For this request codes, the message includes:

XML Tag	Format	Description	Status
Timestamp	Time	Time stamp of triggering event. Used to aid in management of "stale" requests that have timed out. Format is Coordinated Universal Time (e.g. 2005-04-14T13:20:00Z)	Required
ReaderIP	Text	IP address of the embedder or reader. Utilized to determine the correct registration and resolution Router addresses with which to interface.	Required

#### Resolution Response Message

[0326] The Resolution Response Message is the interface that defines the message from a Router in response to data lookup. The Resolution Response interface has primary information, which includes a success code, URL and brief description of the URL, or error code and associated text, and has secondary information which provides content-specific metadata. Examples are shown below, and the format is described below.

#### Resolution Response Root Element, Sub-Elements and Grouping

[0327] All resolution response messages have one root element: <ResolutionResponse> (and </ResolutionResponse>), and sub-elements of <PrimaryInfo> and <SecondaryInfo>. Grouping of resolution requests is allowed (and it is expected to be specified when the transmission method, e.g. SOAP, is finalized). Grouping is useful when the ID Provider reader can read multiple CIDs embedded in the content (e.g. one CID from a content owner and one CID from a retailer).

#### Resolution Response XML Message

[0328] The XML primary response information for all requests is:

XML Tag	Format	Description	Status
Version	Integer	The version of this request message. The current version is 1.	Required
RtnCode	Integer	Return code where "0" means success and any	Required



US 2007/0156726 A1

Jul. 5, 2007

27

-continued

XML Tag	Format	Description	Status
URL	Text	positive number as an error code. For this version, the only valid error code is "1". URL, or list of URLs separated by semicolons ";;"	Required for successful resolution requests, Optional otherwise
Desc	Text	Brief description of URL or error, or list of descriptions of URLs, with each description or error text having 128 characters or less including spaces, and separated by semicolons ";;"	Required, except for Res2ndInfo and ResRegAddress and ResResAddress
BizLogoURL	Text	Content Provider's URL for their company logo.	Optional, except for ResURL, ResURLs, and Res2ndInfo if registered by the Content Provider
BizTemplateURL	Text	Content Provider's URL for their desired display template.	Optional, except for ResURL, ResURLs, and Res2ndInfo if registered by the Content Provider
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

[0329] For the RtnCode, error codes, like "2", may be defined in the future such that the system can automatically handle the error. For this version, providing the vendor or user with the error text is enough, and, thus, "1" is the only valid error code (and "0" or "1" are the only valid RtnCodes).

[0330] The URL Variables are added to the URL after a question mark "?" either as XML data (i.e. between their XML tags) or as text formatted as XML tag=data, both with colons ":" between the XML tags

[0331] XML data example=<IDVersion>1</IDVersion>:CID=999 <CID>999</CID>

[0332] Text example=IDVersion=1:CID=999

[0333] Thus, the URL that is returned can point deep inside a database

[0334] BizLogoURL and BizTemplateURL are required if they were registered by the Content Provider and the request code is ResURL, ResURLs or Res2ndInfo.

[0335] If they were never been registered, they are optional.

[0336] The XML secondary response information for all requests is required for Res2ndInfo request codes or when the Response2ndInfo flag is set:

XML Tag	Format	Description	Status
Title	Text	Title of the content.	Required for 2ndInfo request

-continued

XML Tag	Format	Description	Status
Copyright	Text	Copyright dates.	Required for 2ndInfo request
AdultFlag	Boolean	"1" for adult content, and "0" for all other content	Required for 2ndInfo request
ContentType	Text	Content type choices include: audio - music, audio - speech, video - video track, video - audio track, image and text.	Required for 2ndInfo request
ArtistName	Text	Artist's full name	Required for 2ndInfo request
ArtistEmail	Text	Artist's email address	Required for 2ndInfo request
ArtistPhone	Text	Artist's phone number	Required for 2ndInfo request
ArtistURL	Text	Artist web page.	Required for 2ndInfo request
BizName	Text	Company's full name	Required for 2ndInfo request
BizEmail	Text	Company's email address	Required for 2ndInfo request
BizAddress1	Text	Company's street address, first line	Required for 2ndInfo request
BizAddress2	Text	Company's street address, second line	Required for 2ndInfo request
BizCity	Text	Company's city	Required for 2ndInfo request
BizState	Text	Company's state	Required for 2ndInfo request
BizZip	Text	Company's zip	Required for 2ndInfo request
BizCountry	Text	Company's country	Required for 2ndInfo request
BizPhone	Text	Company's phone number	Required for 2ndInfo request

US 2007/0156726 A1

Jul. 5, 2007

28

-continued

XML Tag	Format	Description	Status
BizURL	Text	Company web page.	Required for 2ndInfo request

[0337] This secondary information comes from a combination of data included when registering a CID and when registering a Content Provider. Although the fields are required for Res2ndInfo or when the Response2ndInfo flag is set, many of the fields are optional when registering and, thus, may not be returned (or be returned blank) even when requested. The secondary information enables the user to view basic artist information, such as may be desirable for photos from a stock agency, and content provider information, which is probably desirable for all content.

#### Resolution Response Message Data

[0338] For the request for URL (ResURL): the message returns a "0" for success in the RtnCode field, the primary URL in the URL field, the brief description of the URL in the Desc field, and the secondary information if requested—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field. The Primary URL has been registered with the Primary flag set, and matches the requested ID Provider ID, ID Version, CID and URL Type.

[0339] For the request for all URLs (ResURLs): the message returns a "0" for success in the RtnCode field, the list of all URLs with the primary URL listed first, separated by semicolons ";", in the URL field, the list of brief descriptions of the URLs, separated by semicolons ";", in the Desc field, and the secondary information if requested—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field. The URLs all match the requested ID Provider ID, ID Version, CID and URL Type. For each URL Type, there is only one primary URL and it is listed first. The secondary information does not have multiple items in each field since it is linked to the CID and not to the URLs. In other words, the secondary info does not change with URLs, only CIDs.

[0340] For the request of secondary information (Res2ndInfo): the message returns a "0" for success in the RtnCode field, and the secondary information in the corresponding fields—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

[0341] For the request of the Router address (ResRegAddress and ResResAddress): the message returns a "0" for success in the RtnCode field, the local Routers' web or IP address in the URL field—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

#### User Display

[0342] When only the primary URL exists, the reader software launches a web browser with the primary URL, such that the primary web page is "immediately" displayed for the user. This always happens with ResURL, and may happen with ResURLs.

[0343] When requesting multiple URLs, the reader software displays the links and descriptions with the Content

Provider's logo and template, if BizLogoURL and BizTemplateURL fields have been registered by the Content Provider. If not, the reader can use its proprietary template. The template defines which secondary CID information (e.g. title, copyright, adult flag content type, artist info and content provider company 10 info) is displayed. The template uses the corresponding XML tags (in between <>) for the secondary data as variables to display this data.

[0344] If the reader is always connected, it should read the CID, request secondary information (potentially caching URLs), and then display to the user that more information is available from <BizName>. This approach causes the response to be immediate upon the user's selection of more information and removes the chance that the user selects more information and no URLs have been registered. In addition, if the content includes multiple CIDs, e.g. one from the content owner and one from the retailer, it also enables the user to differentiate the information source.

#### Router Requirements

[0345] The following requirements enable the Router system to function properly and provide value to vendors and users.

#### CID & URL Requirements

[0346] A Router guarantees that requested or registered CIDs are unique for the given ID Provider ID and ID Version. For requested CIDs, the generating algorithm should guarantee unique IDs. For registered CIDs, the system checks the databases to make sure the CID has not already been registered by that ID Provider with that ID Version. If it has been registered, the system should send back an appropriate error.

[0347] When registering URLs, the system checks that a CID has been registered by the requesting Content Provider name, ID Provider ID and IDVersion. If not, the system should send back an appropriate error.

[0348] A Router guarantees that there is only one Primary URL registered for each URL Type given the ID Provider ID, ID Version and CID.

#### The Databases include:

[0349] The date that a CID is registered (CIDRegDate) and last modified (CIDModDate)

[0350] The date that a URL is registered (URLRegDate) and last modified (URLModDate)

#### Log Files and Usage Reports

[0351] Log file saves the request time, response delay (in milliseconds), request IP address and registration or resolution XML message, excluding private data, for every request. Log files at least span the previous 6 months.

[0352] The aggregate usage data should be visible to the registered ID Providers and Content Providers via an interactive, secure web site, with abilities to export to excel or delimited (tab or comma) text files for download or automatic emailing. Daily, weekly and monthly reports shall be automatically generated for immediate viewing or export. Monthly usage reports should be kept for the life of the system, daily and weekly reports for two years.

US 2007/0156726 A1

Jul. 5, 2007

29

[0353] Reports include:

[0354] 1. For each CID registered by each Content Provider, and for all CIDs registered by each ID Provider and each Content Provider, for the requested date range

[0355] 1.1. For all users and users sending secondary information (including registered users and user sending anonymous secondary information)

[0356] 1.1.1. Total successful links

[0357] 1.1.2. Aggregate links per date (in the date range)

[0358] 1.1.3. Aggregate links per hour for a 24 hour period

[0359] 1.1.4. Aggregate links per day of the week

[0360] 1.1.5. Aggregate links per IP address

[0361] 1.1.6. Average response time

[0362] 1.1.7. Total unsuccessful links grouped by error code

[0363] 1.2. For users sending secondary information

[0364] 1.2.1. Aggregate links per reader ID (which enables ranking of anonymous users)

[0365] 1.2.2. Aggregate links per username (which enables ranking of registered users)

[0366] 1.2.3. Aggregate links per country

[0367] 1.2.4. Aggregate links per zip

[0368] 1.2.5. Aggregate links per sex

[0369] 1.2.6. Aggregate links per age

[0370] 1.2.7. Aggregate links per group

[0371] 1.2.8. Aggregate links per language

[0372] The reports allow Content Providers to access CID specific usage statistics.

[0373] The age groups are defined as:

Age Group Code	Age Group Description
Age0-5	Users between birth and 5 years old
Age6-10	Users between 5 and 10 years old
Age11-15	Users between 11 and 15 years old
Age16-20	Users between 16 and 20 years old
Age21-25	Users between 21 and 25 years old
Age26-30	Users between 26 and 30 years old
Age31-40	Users between 31 and 40 years old
Age41-50	Users between 41 and 50 years old
Age51-60	Users between 51 and 60 years old
Age61-80	Users between 61 and 80 years old
Age81+	Users over 81 years old

[0374] The registered vendors shall also be able to create custom reports with any begin and end data in the last 2 years. These custom reports will take a little time to calculate since daily reports have to be analyzed. The system may allow for custom reports with fields not included in the report list below for the last 6 months. These reports will take some time to calculate since they require log files to be analyzed.

Distributed Architecture

[0375] In order for the distributed architecture of the Router to function properly and be able to be expanded in the future, the following occur.

[0376] The reader requests the web or IP address of registration and/or resolution router, which ever are applicable, every week. Thus, the system architecture can be dynamically changed every week.

[0377] The number of Mirror Routers is set by the Registration Authority. The Mirror Routers forward all registration requests immediately to the Central Router, and wait its response that the information is correct (or the CID is unique). They also send aggregate usage files to the Central Router and synchronize with the Central Database daily. The daily time is not sent so that they don't all hit the Central Router simultaneously. Finally, they forward resolution requests for CIDs that don't exist in their database and wait for the response before responding to the User, to make sure they were not registered during that recent day.

XML Schemas

[0378] The XML schemas are specified, most likely, with grouped (e.g., within WSDL) as follows:

[0379] Registration CID Request Message Schema for CreateCID and RegCID request codes

[0380] One for integer and one for text CIDs

[0381] Registration URL Request Message Schema for RegURL, DelURL, RegURLs, and DelURLs request codes

[0382] One for integer and one for text CIDs

[0383] Registration Contact Information Request Message Schema for UpdateContentProvider, UpdateIDProvider, RegUser and UpdateUser request codes

[0384] Registration Response Message Schema as used by all registration requests

[0385] Resolution Request Message Schema for ResURL, ResURLs, and Res2ndInfo request codes

[0386] One for integer and one for text CIDs

[0387] Resolution Request Message Schema for Request Router addresses via ResRegAddress and Res-ResAddress

[0388] Resolution Response Message Schema as used by all resolution requests

Packing and Transmission Methods

[0389] The XML packing and transmission methods need to be specified, and include http, https, WSDL, SOAP and secure Web Services, such as SAML, WS-License or WS-Security. Public key architecture, such as described in XKMS, X-KRSS, and X-KISS, is probably not required since this example implementation has Content Provider Name and password as part of the standard XML message interface, and the contact data is not mission critical.

[0390] Importantly, the http or https link should be maintained from request to response, and only broken after the response is received, thus increasing efficiency and reducing

US 2007/0156726 A1

Jul. 5, 2007

30

risk of firewall interfering with the response (as well as not requiring the client embedder or reader software to act as a server).

#### Database Elements

[0391] Describing the database elements and possible arrangements helps the reader understand the interface and router since it provides an overview of how the requests are handled.

#### Database Management

[0392] Database management is desirable and includes:

[0393] Check URLs at least once a month

[0394] Report dead URLs to Content Provider (using contact person's email) whom requested the CID and registered the related URL

[0395] Uses CID Owner Information

[0396] Report CIDs with no Primary URL but with additional URLs to Content Provider

[0397] This assumes that a CID with no URLs at all has not been distributed yet or is not used for that URL Type

#### Database Information

[0398] The Primary Databases include the following information. The structure does not need to be as implied by the tables. However, the tables provide a nice outline and help to conceptually understand how the message interface is used.

[0399] CID Owner Information

Label	ContentProviderName	IDProviderID	IDVersion	CID	CIDExpiration	CIDRegDate	CIDModDate
Type	Text	Integer	Integer	Integer or Text	Date	Date	Date

Label	Title	Copyright	AdultFlag	ContentType	ArtistName	ArtistEmail	ArtistPhone	ArtistURL
Type	Text	Text	Boolean	Text	Text	Text	Text	Text

[0400] This informatin is entered when requesting a CID (CreatCID) or when registering pre-existing CID from another ID standard (RegCID).

[0401] the CID for RegCID is checked for uniqueness given ID Provider ID and ID Version

[0402] CID Link Information

Label	IDProviderID	IDVersion	CID	Primary	URL Type	URL	URLVariable	Desc
Type	Integer	Integer	Integer or Text	Boolean	Byte	Text	Text	Text

Label	URLActivation	URLExpiration	URLRegDate	URLModDate
Type	Date	Date	Date	Date

[0403] This information is entered or removed when a URL is registerd or deleted (RegURL and RegURLs or DeURL and DeURLs), respectively

[0404] Check that only one Primary URL for each URL Type is registered when registering a primary URL with RegURL (as described in the spec)

[0405] There may be one database or several databases

[0406] There could be a different database for each ID Provider ID, or ID Provider ID and ID Version.

[0407] The database could combine ID Provider ID, ID Version, and CID to one index.

[0408] There could be two databases, one for numeric combinations (where CID is an integer) and one for text combinations (where CID is text based) or one database with a text index. The registration or resolution schema could be used to determine which database to use. The integer combination is optimal for efficiency.

[0409] Fields may be combined.

[0410] For example, rather than having URLType, URI and Description, there may be FullURL, FullDesc, WAPURL, WAPDesc, WMIURL, WMIDesc-or any combination.

[0411] The database can convert URL Type text to integer for speed

[0412] URL Type: Full=1, a WAP=2 and a WMI=3.

US 2007/0156726 A1

Jul. 5, 2007

31

The Secondary Databases include the following information:

[0413] Content Provider, ID Provider and User Information

Label	<RegName>*	Password	BizName	BizEmail	BizAddress1	BizAddress2	BizCity
Type	<type>	Text	Text	Text	Text	Text	Text

Label	BizState	BizZip	BizCountry	BizPhone	BizURL	BizLogoURL	BizTemplateURL
Type	Text	Text	Text	Text	Text	Text	Text

Label	Name	Email	Address1	Address2	City	State	Zip	Country	Phone	Cell	Fax
Type	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text	Text

Label	IM	Language	Sex	Age
Type	Text	Text	Text	Integer

<RegName> is ContentProviderName, IDProviderID, or UserName  
 <type> is Text, Integer, or Text for ContentProvider, ID Provider or User databases, respectively  
 Boolean can be stored with TRUE = 1 and FALSE = 0  
 Sex can be stored as Boolean with Female = TRUE = 1 and Male = FALSE = 0

#### Registration Message Examples

[0414] Create a Content ID

```

Request Message (CreateCID)
<RegistrationCIDRequest>
  <Header>
    <Version>1</Version>
    <IDProviderID>123</IDProviderID>
    <RequestCode>CreateCID</RequestCode>
  </Header>
  <Body>
    <PrimaryInfo>
      <ContentProviderName>Disney</ContentproviderName>
      <Password>walt4pres</Password>
      <IDVersion>1</IDVersion>
    </PrimaryInfo>
    <SecondaryInfo>
      <Title>Fantasia</Title>
      <Copyright>1960</Copyright>
      <AdultFlag>0</AdultFlag>
      <ContentType>video-videotrack</ContentType>
      <ArtistEmail>fantasia@disney.com</ArtistEmail>
    </SecondaryInfo>
  </Body>
</RegistrationCIDRequest>
Response Message (CreateCID)
<RegistrationResponse>
  <Version>1</Version>
  <RtnCode>0</RtnCode>
  <Desc>999</Desc>
</RegistrationResponse>
Register a Pre-Existing Content ID
Request Message (RegCID)
<RegistrationCIDRequest>
  <Header>
    <Version>1</Version>
    <IDProviderID>321</IDProviderID>
    <RequestCode>RegCID</RequestCode>
  </Header>
  <Body>
    <PrimaryInfo>
      <ContentProviderName>Apple</ContentproviderName>
      <Password>1984year</Password>
      <IDVersion>2</IDVersion>

```

-continued

```

  <CID>111</CID>
</PrimaryInfo>
<SecondaryInfo>
  <Title>Fantasia</Title>
  <Copyright>1960</Copyright>
  <AdultFlag>0</AdultFlag>
  <ContentType>video-audiotrack</ContentType>
</SecondaryInfo>
</Body>
</RegistrationCIDRequest>
Response Error Message (RegCID)
<RegistrationResponse>
  <Version>1</Version>
  <RtnCode>1</RtnCode>
  <URL>http://www.cmds.com/error/error8.html</URL>
  <Desc>
    CID is already registered. Please verify the CID and ID
    Version and try again.
  </Desc>
</RegistrationResponse>

```

#### Register multiple URLs

[0415] Registering one URL with RegURL is extremely similar except there is only one data element in the Primary, URLType, URL, URLVariables, URLActivation, URLExpriation, and Desc fields. Alternatively, multiple URLs can be registered by grouping multiple RegURL calls in the transmission. Deleting one or more URLs with DelURL or DelURLs is very similar except that URLVariables, URLActivation, URLExpriation, and Desc are optional, and ignored if included. As for RegURL, in DelURL, there is only one URL included in the URL XML field.

```

Request Message (RegURLs)
<RegistrationURLRequest>
  <Header>
    <Version>1</Version>

```

US 2007/0156726 A1

Jul. 5, 2007

32

-continued

---

```

<IDProviderID>123</IDProviderID>
<RequestCode>RegURLs</RequestCode>
</Header>
<Body>
  <ContentProviderName>Disney</ContentProviderName>
  <Password>walt4pres</Password>
  <IDVersion>1</IDVersion>
  <CID>999</CID>
  <Primary>1;1;0;0;0</Primary>
  <URLType>Full;WAP;Full;WAP;Full;WAP</URLType>
  <URL>
    www.disney.com/fantasia;
    wap.disney.com/fantasia;
    www.disney.com/fantasia/music;
    wap.disney.com/fantasia/music;
    www.amazon.com/search?fantasia;
    wap.amazon.com/search?fantasia;
  </URL>
  <URLVariables>
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
    IDProviderID:IDVersion:CID:ReaderID:OS:Username;
  </URLVariables>
  <Desc>
    Fantasia movie info from Disney.com;
    Fantasia movie info from Disney.com (WAP Format);
    Fantasia music info from Disney.com;
    Fantasia music info from Disney.com (WAP Format);
    Fantasia memorabilia from Amazon.com;
    Fantasia memorabilia from Amazon.com (WAP format);
  </Desc>
</Body>
</RegistrationURLRequest>
Response Message (RegURLs)
<RegistrationResponse>
  <Version>1</Version>
  <RtnCode>0</RtnCode>
</RegistrationResponse>

```

---

### Register a User

[0416] Updating a Content Provider, ID Provider or User is very similar to registering a User, except that the name and password have already been assigned as opposed to being checked for uniqueness. In addition, the User does not need to enter business information.

---

```

Request Message (RegUser)
<RegistrationContactInfoRequest>
  <Header>
    <Version>1</Version>
    <IDProviderID>123</IDProviderID>
    <RequestCode>RegUser</RequestCode>
  </Header>
  <Body>
    <RegName>klevy</RegName>
    <Password>ken1sgreat</Password>
    <BizName>AIPL</BizName>
    <BizEmail>levy@aipl.com</BizEmail>
    <BizAddress1>110 NE Cedar Street</BizAddress1>
    <BizCity>Stevenson</BizCity>
    <BizState>WA</BizState>
    <BizZip>98648</BizZip>
    <BizCountry>USA</BizCountry>
    <BizPhone>509-427-5374</BizPhone>
    <BizURL>www.AIPL.com</BizURL>
    <NameFirst>Ken</NameFirst>
    <NameLast>Levy</NameLast>

```

---

-continued

---

```

  <Email>levy@aipl.com</Email>
  <Address1>110 NE Cedar Street</Address1>
  <City>Stevenson</City>
  <State>WA</State>
  <Zip>98648</Zip>
  <Country>USA</Country>
  <Cell>509-427-5374</Cell>
  <Language>English</Language>
  <Sex>M</Sex>
  <Age>40</Age>
</Body>
</RegistrationContactInfoRequest>

```

---

### Response Message (RegUser)

[0417] First, the user receives an error because their user-name “klevy” exists.

---

```

<RegistrationResponse>
  <Version>1</Version>
  <RtnCode>1</RtnCode>
  <URL>http://www.cmds.com/error/error6.html</URL>
  <Desc>
    Username is already registered.
    Please try a different username.
  </Desc>
</RegistrationResponse>

```

---

[0418] Then, they re-try with the user name “kenlevy”, and it is successful.

---

```

<RegistrationResponse>
  <Version>1</Version>
  <RtnCode>0</RtnCode>
  <Desc>kenlevy</Desc>
</RegistrationResponse>

```

---

### Resolution Message Examples

#### Resolve URLs

[0419] Resolving one URL is very similar, where the only change is that one URL is returned in the response message.

---

```

Request Message (ResURLs)
<ResolutionURLRequest>
  <Header>
    <Version>1</Version>
    <IDProviderID>123</IDProviderID>
    <RequestCode>ResURLs</RequestCode>
  </Header>
  <Body>
    <PrimaryInfo>
      <Timestamp>2005-04-14T13:20:00Z</Timestamp>
      <IDVersion>1</IDVersion>
      <CID>999</CID>
      <URLType>Full</URLType>
      <Response2ndInfo>TRUE</Response2ndInfo>
    </PrimaryInfo>
    <SecondaryInfo>
      <ReaderID>789</ReaderID>
      <TransactionID>235</TransactionID>
      <OS>Windows</OS>

```

---



US 2007/0156726 A1

Jul. 5, 2007

33

-continued

---

```

    <Username>kenlevy</Username>
  </SecondaryInfo>
</Body>
</ResolutionURLRequest>
Response Message (ResURLs)
<ResolutionResponse>
  <PrimaryInfo>
    <Version>1</Version>
    <RtnCode>0</RtnCode>
    <URL>
      www.disney.com/fantasia?
        IDProviderID=123:IDVersion=1:CID=999;
        ReaderID=789:ReaderID=789:OS=Windows;
        Username=kenlevy;
      www.disney.com/fantasia/music?
        IDProviderID=123:IDVersion=1:CID=999;
        ReaderID=789:ReaderID=789:OS=Windows;
        Username=kenlevy;
      www.amazon.com/search?fantasia?
        IDProviderID=123:IDVersion=1:CID=999;
        ReaderID=789:ReaderID=789:OS=Windows;
        Username=kenlevy;
    </URL>
    <Desc>
      Fantasia movie info from Disney.com;
      Fantasia music info from Disney.com;
      Fantasia memorabilia from Amazon.com;
    </Desc>
    <BizLogoURL>disney.com/CMDS/logo.jpg</BizLogoURL>
    <BizTemplateURL>disney.com/CMDS/template</BizTemplateURL>
  </PrimaryInfo>
  <SecondaryInfo>
    <Title>Fantasia</Title>
    <Copyright>1960</Copyright>
    <AdultFlag>0</AdultFlag>
    <ContentType>video-videotrack</ContentType>
    <ArtistEmail>fantasia@disney.com</ArtistEmail>
    <BizName>Disney</BizName>
    <BizURL>www.disney.com</BizURL>
  </SecondaryInfo>
</ResolutionResponse>

```

---

## Resolve Router Address

[0420] Requesting the registration router address is very similar, where only the request code changes.

---

```

Request Message (ResResAddress)
<ResolutionAddressRequest>
  <Header>
    <Version>1</Version>
    <IDProviderID>123</IDProviderID>
    <RequestCode>ResResAddress</RequestCode>
  </Header>
  <Body>
    <Timestamp>2005-04-14T13:20:00Z</Timestamp>
    <ReaderIP>206.58.236.61</ReaderIP>
  </Body>
</ResolutionAddressRequest>
Response Message (ResResAddress)
<ResolutionResponse>
  <PrimaryInfo>
    <Version>1</Version>
    <RtnCode>0</RtnCode>

```

-continued

---

```

    <URL>198.70.207.6/CMDS/cgi_bin</URL>
  </PrimaryInfo>
</ResolutionResponse>

```

---

## Error Text Examples

[0421] Some error text examples include:

[0422] 1. "Content is registered, but no URL in database. Please contact <ContentProviderName> at <BizEmail>."

[0423] 2. "Content is registered, but URL is marked as inactive. Please contact <ContentProviderName>."

[0424] 3. "No record in database matching the content. Please contact <ContentProviderName> at <BizEmail>."

[0425] 4. "Request format error—incomplete data. Please contact software provider."

[0426] 5. "Primary URL is already registered. Please try again and verify settings, especially Primary and IDVersion."

[0427] 6. "Username is already registered. Please try a different username."

[0428] 7. "Password is not a valid format. It needs to be at least 6 characters with a number. Please try again."

[0429] 8. "CID is already registered. Please verify the CID and ID Version and try again."

[0430] 9. "CID is not registered. Please verify the Content Provider Name, ID Version and CID."

[0431] 10. "CID is expired as content is out of date."

[0432] The system is very flexible and can enable multiple future possibilities. The enhancements include: (1) seed and interoperate multiple ID systems, and (2) enabling "buy now" links for illegitimate content. Another enhancement includes integrating multiple ID Provider technologies or cached Routers.

## Seed and Interoperate with Multiple ID Systems

[0433] In a future version of the messages (e.g. with the number "2" in the Version XML header tag), the following steps could enable this system to seed and interoperate with multiple ID systems:

[0434] 1. Add a registration request code to register other IDs that are linked to a CID

[0435] 2. Add a resolution request code to return the other IDs

[0436] 3. Add a resolution request code to return the CID given the other ID

[0437] Even when the other IDs are returned, the receiving software needs to know how to register the other IDs with each proprietary system to enable seeding multiple systems from one piece of software.

US 2007/0156726 A1

Jul. 5, 2007

34

## Registration Request Message

[0438] Specifically, the following request code could be added to version 2 of the registration message:

[0439] Request Code: RegOtherIDs

XML Tag	Format	Description	Status
ContentProviderName	Integer	Content provider name.	Required
Password	Text	Content provider password.	Required
IDVersion	Integer	The ID version provides the version of the CID algorithm.	Required
CID	Text	The unique content ID formatted according to the IDVersion. Since these requests are few, speed is not critical and integer CIDs are carried as text.	Required
W3C_URI	Text	World Wide Web Consortium (W3C) Uniform Resource Identifier (URI). <a href="http://www.ietf.org/rfc/rfc2396.txt">http://www.ietf.org/rfc/rfc2396.txt</a> . Format is text.	Optional
IDF_DOI	Text	International DOI Foundation (IDF) Digital Object Identifier (DOI). <a href="http://www.doi.org">www.doi.org</a> . Format is text.	Optional
OASIS_ERI	Text	OASIS's unique ID. <a href="http://www.oasis-open.org/who/">http://www.oasis-open.org/who/</a> . Format is text.	Optional
CISAC_CIS	Integer	International Confederation Of Societies Of Authors And Composers (CISAC) Common Information System (CIS). <a href="http://www.CISAC.org">http://www.CISAC.org</a> . Format is 96 bits.	Optional
ISO_ISRC	Text (12 chars)	The ISRC (International Standard Recording Code) was developed by ISO (International Organisation for Standardisation) to identify sound and audio-visual recordings. It is known as International Standard ISO 3901. <a href="http://www.iso.ch/cate/d9515.html">http://www.iso.ch/cate/d9515.html</a> . Format is 12 characters.	Optional
SMPTE_UMID	Integer	Society of Motion Picture Television Engineers (SMPTE) Unique Material Identifier (UMID). <a href="http://www.smpete.org">www.smpete.org</a> 330M-2000. Format is 64 bits.	Optional
ISO-IEC_MPEG21_DII	Text	Moving Pictures Expert Group (MPEG) Digital Item Identification (DII). ISO/IEC 21000-3. Format is text.	Optional
ISO_ISAN	Integer	ISO International Standard Audiovisual Number (ISAN). <a href="http://www.nlc-bnc.ca/iso/tc46sc9/isan.htm">http://www.nlc-bnc.ca/iso/tc46sc9/isan.htm</a> . Format is 64 bits.	Optional
ISO_V-ISAN	Integer	ISO Identifier for Versions of Audiovisual works. w4636. Format is 96 bits.	Optional
ISO_ISWC	Text (9 chars)	ISO Information System Work Code (ISWC). <a href="http://www.iswc.org">http://www.iswc.org</a> . Format is 9 digits.	Optional
cIDf	Integer	Content ID Forum (cIDf). <a href="http://www.cidf.org">www.cidf.org</a> . Format is 96 bits.	Optional
IFPI_Grid	Text (18 chars)	IFPI Global Release Identifier (GRid). <a href="http://www.ifpi.org/grid">www.ifpi.org/grid</a> . The format is 18 characters.	Optional
EBU_WUMI	Integer	European Broadcast Union (EBU) Watermark Unique Material Identifier (WUMI). <a href="http://www.ebu.ch">http://www.ebu.ch</a> . Format is 64 bits.	Optional

US 2007/0156726 A1

Jul. 5, 2007

35

-continued

XML Tag	Format	Description	Status
Ad-ID	Text (12 chars)	Ad-ID LLC's advertising ID. www.aaaa.org. Format is 12 characters.	Optional
TVAnyTime_CRID	Text	TV Anytime's unique identifier. http://www.tv-anytime.org. Format is text.	Optional
CRF	Text	Content Reference Forum (CRF). www.crforum.org. Format is text.	Optional
ISO_ISBN	Integer (10 digits)	ISO International Standard Book Number (ISBN). Format is 10 digits.	Optional
ISO_ISSN	Integer (8 digits)	ISO International Standard Serial Number (ISSN). Format is 8 digits.	Optional
ISO_ISTC	Text	ISO International Standard Textual Work Code. Format is text (since not sure of format).	Optional
ONIX	Text	ONline Information eXchange (ONIX) http://www.editeur.org/onix.html. Format is text (since not sure of format).	Optional
UCC_UPC	Integer (12 digits)	Uniform Code Council (UCC) Universal Product Code (UPC) www.uc-council.org. Format is 12 digits.	Optional
EPCGlobal_EPC	Integer	EPCGlobal Electronic Product Code (EPC). www.EPCGlobal.com. Format is 96 bits.	Optional
Symbol_PDF417	Array (1.1 kBytes)	Symbol 2D bar code standard. Portable Document Format (PDF). Format is 1.1 kBytes.	Optional
Private	Binary	Binary data that can be used by the ID Provider for any private reason.	Optional

## Registration Response Message

[0440] For the request for registering other system ID (RegOtherIDs): the message returns a "0" for success in the RtnCode field—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

## Resolution Request Message

[0441] The following resolution request codes could be added, with the corresponding response data:

Request Code: ResCIDGivenOtherID

[0442] This request uses the same format as used in ResURL, except the CID XML field contains the other ID data (and the CID is returned).

Request Code: ResOtherIDs

[0443] This request includes primary information, and uses the same data as for RegOtherIDs, except that the Content Provider ID and Password XML tags are optional. In addition, the Timestamp XML tag from the ResURL Request Code above is included as required fields.

## Resolution Response Message

[0444] For requesting a CID given another unique ID (ResCIDGivenOtherID): the message returns a "0" for success in the RtnCode field, the CID in the Desc field, and the secondary information if requested—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

[0445] For requesting other unique IDs (ResOtherIDs): the message returns a "0" for success in the RtnCode field, and a list of the registered other IDs with each entry separated by a semicolon ";" each entry in the format consisting of the other system XML tag as defined for RegOtherIDs request code and related ID separated by a colon ":" in the Desc field, and the secondary information in the corresponding fields if requested—or "1" for error in the RtnCode, the error URL in the URL field, and the error text in the Desc field.

## Database Elements

[0446] The corresponding database elements could be:

[0447] Unique ID Systems Database

Label	IDProviderID	Version	CID	W3C URI	IDF DOI	OASIS ERI
Size	Integer	Integer	Integer or Text	Text	Text	Text

US 2007/0156726 A1

Jul. 5, 2007

36

-continued

Label	CISAC CIS	ISRC	SMPTE UMID	ISO/IEC DII	MPEG21	ISO ISAN	ISO VISAN
Size	Integer (96 bits)	Text (12 chars)	Integer (64 bits)	Text		Integer (64 bits)	Integer (96 bits)
Label	ISO ISWC	cIDf	IFPI Grid	EBU WUMI	Ad-ID	TV AnyTime CRID	CRF
Size	Text (9 digits)	Integer (96 bits)	Text (18 chars)	Integer (64 bits)	Text (12 chars)	Text	Text
Label	ISO ISBN	ISO ISSN	ISO ISTC	ONIX	UCC UPC	UCC EPC	Symbol PDF 417
Size	Integer (10 digits)	Integer (8 digits)	Text?	Text?	Integer (12 digits)	Integer (96 bits)	Array (1.1 kBytes)

#### “Buy Now” for Illegitimate Content

[0448] When the multimedia player goes to play or transfer raw content (i.e. neither encrypted nor digitally signed), it calls the reader plug-in to check for a CID and whether or not the content is copy protected. If this raw content is copy protected and contains a CID, an explanation and “buy now” link can be provided to the user.

[0449] Note that the existence of a CID does not necessarily mean that the content cannot be played, as the consumer may have converted a purchased CD to compressed audio, or bought non-encrypted content. The content needs to also have a copy protection identifier, such as Copy Control Information (CCI) in the content ID, such as DWM, payload. The Player can log this event so it only checks raw content once.

[0450] By including a rights’ flag linked to URLs, one primary rights URL can be registered using existing registration requests, and the rights URL can be returned to the reader to offer a “buy now” link to purchase legitimate content. Specifically, the rights flag is an additional XML field in registration and resolution messages, and is included in the database elements with the URLs.

#### Universal Reader Interface

[0451] When content owners and retailers are both acting as Content Providers, thus, both marking the content with their own CID, the user sees two different “more info” buttons, one from the content owner and one from the retailer. There could even be an additional “more info” button from the distributor, such as ISP.

[0452] The Registration Authority could, in the future, provide a plug-in that provides a universal reader interface that integrates CIDs and URLs from different ID Provider detectors. This example implementation could be expanded to include a standard reader-application interface, which is the interface between a generic reader plug-in and ID Provider detector. Thus, all installed ID technology detectors could call (or be called from) the generic reader plug-in, and all CIDs sent to a Router from the generic reader plug-in.

[0453] Not only would the interface be specified for this generic reader plug-in, but the Registration Authority would also have to provide the plug-in for distribution for all multimedia players. The generic plug-in could have the ID

provider plug-ins call it with their resolution request XML messages, then display one “more info” button for the user, and finally send requests for all detected CIDs to a Router. This approach would only require the ID Provider plug-ins to change their call from an IP address to an internal DLL, and remove their user display.

#### Cached Routers

[0454] The architecture enables Mirror Routers to link to Cached Routers for resolution requests. Cached Routers are not applicable to registration requests since these requests occur neither often nor repetitively, and are immediately forwarded to the Central Router. These Cached Routers will temporarily maintain their Cached Database for recent resolution requests. It is expected that the data will have a time limit set by the Registration Authority. The Cached Routers will also send usage information to their linked Mirror Router daily. This architecture becomes truly distributed and efficient.

#### CONCLUDING REMARKS

[0455] Having described and illustrated the principles of the technology with reference to specific implementations, it will be recognized that the technology can be implemented in many other, different, forms. To provide a comprehensive disclosure without unduly lengthening the specification, applicants incorporate by reference the patents and patent applications referenced above.

[0456] The methods, processes, and systems described above may be implemented in hardware, software or a combination of hardware and software. For example, the auxiliary data encoding processes may be implemented in a programmable computer or a special purpose digital circuit. Similarly, auxiliary data decoding may be implemented in software, firmware, hardware, or combinations of software, firmware and hardware. The methods and processes described above may be implemented in programs executed from a system’s memory (a computer readable medium, such as an electronic, optical or magnetic storage device).

[0457] The particular combinations of elements and features in the above-detailed embodiments are exemplary only; the interchanging and substitution of these teachings with other teachings in this and the incorporated-by-reference patents/applications are also contemplated. The inven-

US 2007/0156726 A1

Jul. 5, 2007

37

tor submits that the invention encompasses the claims set forth below, as well as systems and computer readable mediums that implement these methods. The embodiments described in this document are also inventive and applicant reserves the right to claim various embodiments and combinations thereof as its invention.

I claim:

1. A method of associating a content object with metadata comprising:

receiving a content identifier for a content object from among a set of content identifiers;

providing a unique bounding identifier for the set of content identifiers;

using a combination of the content identifier and the unique bounding identifier to form a globally unique identifier for the content object; and

associating the globally unique identifier with a metadata source to enable routing of an entity that supplies the globally unique identifier to the metadata source.

2. The method of claim 1 wherein associating the globally unique identifier with the metadata source comprises storing a metadata source identifier that identifies the metadata source located at another network location.

3. The method of claim 2 wherein the metadata source identifier comprises a URL of the metadata source.

4. The method of claim 1 wherein the set of content identifiers is pre-assigned by an ID provider and registered with a directory by providing the unique bounding identifier to the set of content identifiers.

5. The method of claim 1 wherein a directory assigns both the content identifier for the content object and the unique bounding identifier for the ID provider and provides the content identifier to an ID provider for inserting the content identifier in the content object.

6. The method of claim 5 wherein the directory utilizes the unique bounding identifier to determine the content identifier.

7. The method of claim 1 wherein a first set of content identifiers is pre-assigned by a first ID provider and registered with a directory by providing a first unique bounding identifier to the first set of content identifiers, and wherein the directory assigns a second set of content identifiers for a second ID provider, which in turn, inserts the second set of content identifiers in corresponding content objects.

8. The method of claim 1 wherein the unique bounding identifier comprises an ID provider identifier.

9. The method of claim 8 wherein the unique bounding identifier further comprises an ID version.

10. The method of claim 1 wherein the combination of the content identifier and the unique bounding identifier are provided by a user to a directory, which in turn, routes the user to the metadata source associated with the globally unique identifier for the content object.

11. The method of claim 1 wherein the metadata source is provided by the content provider of the content object and is represented as at least a first link to metadata of the content provider and at least a second link to metadata of another participant.

12. The method of claim 11 wherein the metadata of another participant includes a link to an online commerce transaction opportunity relating to the content object.

13. The method of claim 11 wherein the content provider is a content owner of the content object.

14. The method of claim 1 wherein said content object is associated with a second, different metadata source, the method comprising:

receiving a second content identifier for said content object from among a second set of content identifiers;

providing a second unique bounding identifier for the second set of content identifiers;

using a second combination of the second content identifier and the second unique bounding identifier to form a second globally unique identifier for said content object; and

associating the second globally unique identifier with a metadata source to enable routing of an entity that supplies the globally unique identifier to the metadata source.

15. A computer readable medium on which is stored instructions for performing the method of claim 1

16. A system for associating a content object with metadata comprising:

a registration interface receiving a content identifier for a content object from among a set of content identifiers, and for providing a unique bounding identifier for the set of content identifiers; and

a database operable to use a combination of the content identifier and the unique bounding identifier to form a globally unique identifier for the content object, and to associate the globally unique identifier with a metadata source to enable routing of an entity that supplies the globally unique identifier to the metadata source.

17. The system of claim 16 further including a router operable to receive a content identifier, and in response to use the bounding identifier in combination with the content identifier to look up the corresponding metadata source in the database.

18. The system of claim 17 wherein the router provides a metadata source identifier to a requesting entity, which in turn, uses the metadata source identifier to establish a connection with the metadata source and obtain metadata associated with the content object.

\* \* \* \* \*

# EXHIBIT 8





# Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

OASIS Standard, 15 March 2005

## Document identifier:

saml-core-2.0-os

## Location:

<http://docs.oasis-open.org/security/saml/v2.0/>

## Editors:

Scott Cantor, Internet2  
John Kemp, Nokia  
Rob Philpott, RSA Security  
Eve Maler, Sun Microsystems

## SAML V2.0 Contributors:

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rebekah Metz, Booz Allen Hamilton  
Rick Randall, Booz Allen Hamilton  
Thomas Wisniewski, Entrust  
Irving Reid, Hewlett-Packard  
Paula Austel, IBM  
Maryann Hondo, IBM  
Michael McIntosh, IBM  
Tony Nadalin, IBM  
Nick Ragouzis, Individual  
Scott Cantor, Internet2  
RL 'Bob' Morgan, Internet2  
Peter C Davis, Neustar  
Jeff Hodges, Neustar  
Frederick Hirsch, Nokia  
John Kemp, Nokia  
Paul Madsen, NTT  
Steve Anderson, OpenNetwork  
Prateek Mishra, Principal Identity  
John Linn, RSA Security  
Rob Philpott, RSA Security  
Jahan Moreh, Sigaba  
Anne Anderson, Sun Microsystems

Eve Maler, Sun Microsystems  
Ron Monzillo, Sun Microsystems  
Greg Whitehead, Trustgenix

**Abstract:**

This specification defines the syntax and semantics for XML-encoded assertions about authentication, attributes, and authorization, and for the protocols that convey this information.

**Status:**

This is an **OASIS Standard** document produced by the Security Services Technical Committee. It was approved by the OASIS membership on 1 March 2005.

Committee members should submit comments and potential errata to the [security-services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located at [http://www.oasis-open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security). The committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog of any changes made to this document as a result of comments.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the Security Services TC (<http://www.oasis-open.org/committees/security/ipr.php>).

# Table of Contents

61	1 Introduction.....	7
62	1.1 Notation.....	7
63	1.2 Schema Organization and Namespaces.....	8
64	1.3 Common Data Types.....	8
65	1.3.1 String Values.....	8
66	1.3.2 URI Values.....	9
67	1.3.3 Time Values.....	9
68	1.3.4 ID and ID Reference Values.....	9
69	2 SAML Assertions.....	11
70	2.1 Schema Header and Namespace Declarations.....	11
71	2.2 Name Identifiers.....	12
72	2.2.1 Element <BaseID>.....	12
73	2.2.2 Complex Type NameIDType.....	13
74	2.2.3 Element <NameID>.....	14
75	2.2.4 Element <EncryptedID>.....	14
76	2.2.5 Element <Issuer>.....	15
77	2.3 Assertions.....	15
78	2.3.1 Element <AssertionIDRef>.....	15
79	2.3.2 Element <AssertionURIRef>.....	15
80	2.3.3 Element <Assertion>.....	15
81	2.3.4 Element <EncryptedAssertion>.....	17
82	2.4 Subjects.....	17
83	2.4.1 Element <Subject>.....	18
84	2.4.1.1 Element <SubjectConfirmation>.....	18
85	2.4.1.2 Element <SubjectConfirmationData>.....	19
86	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	20
87	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
88	2.5 Conditions.....	21
89	2.5.1 Element <Conditions>.....	21
90	2.5.1.1 General Processing Rules.....	22
91	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
92	2.5.1.3 Element <Condition>.....	23
93	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
94	2.5.1.5 Element <OneTimeUse>.....	24
95	2.5.1.6 Element <ProxyRestriction>.....	25
96	2.6 Advice.....	25
97	2.6.1 Element <Advice>.....	26
98	2.7 Statements.....	26
99	2.7.1 Element <Statement>.....	26
100	2.7.2 Element <AuthnStatement>.....	26
101	2.7.2.1 Element <SubjectLocality>.....	28
102	2.7.2.2 Element <AuthnContext>.....	28
103	2.7.3 Element <AttributeStatement>.....	29
104	2.7.3.1 Element <Attribute>.....	29
105	2.7.3.1.1 Element <AttributeValue>.....	30
106	2.7.3.2 Element <EncryptedAttribute>.....	31
107	2.7.4 Element <AuthzDecisionStatement>.....	31
108	2.7.4.1 Simple Type DecisionType.....	33
109	2.7.4.2 Element <Action>.....	33

110	2.7.4.3 Element <Evidence>.....	34
111	3 SAML Protocols.....	35
112	3.1 Schema Header and Namespace Declarations.....	35
113	3.2 Requests and Responses.....	36
114	3.2.1 Complex Type RequestAbstractType.....	36
115	3.2.2 Complex Type StatusResponseType.....	38
116	3.2.2.1 Element <Status>.....	39
117	3.2.2.2 Element <StatusCode>.....	39
118	3.2.2.3 Element <StatusMessage>.....	42
119	3.2.2.4 Element <StatusDetail>.....	42
120	3.3 Assertion Query and Request Protocol.....	42
121	3.3.1 Element <AssertionIDRequest>.....	42
122	3.3.2 Queries.....	42
123	3.3.2.1 Element <SubjectQuery>.....	42
124	3.3.2.2 Element <AuthnQuery>.....	43
125	3.3.2.2.1 Element <RequestedAuthnContext>.....	44
126	3.3.2.3 Element <AttributeQuery>.....	45
127	3.3.2.4 Element <AuthzDecisionQuery>.....	46
128	3.3.3 Element <Response>.....	46
129	3.3.4 Processing Rules.....	47
130	3.4 Authentication Request Protocol.....	47
131	3.4.1 Element <AuthnRequest>.....	48
132	3.4.1.1 Element <NameIDPolicy>.....	50
133	3.4.1.2 Element <Scoping>.....	51
134	3.4.1.3 Element <IDPList>.....	52
135	3.4.1.3.1 Element <IDPEntry>.....	52
136	3.4.1.4 Processing Rules.....	53
137	3.4.1.5 Proxying.....	54
138	3.4.1.5.1 Proxying Processing Rules.....	54
139	3.5 Artifact Resolution Protocol.....	55
140	3.5.1 Element <ArtifactResolve>.....	56
141	3.5.2 Element <ArtifactResponse>.....	56
142	3.5.3 Processing Rules.....	56
143	3.6 Name Identifier Management Protocol.....	57
144	3.6.1 Element <ManageNameIDRequest>.....	57
145	3.6.2 Element <ManageNameIDResponse>.....	58
146	3.6.3 Processing Rules.....	58
147	3.7 Single Logout Protocol.....	59
148	3.7.1 Element <LogoutRequest>.....	60
149	3.7.2 Element <LogoutResponse>.....	60
150	3.7.3 Processing Rules.....	61
151	3.7.3.1 Session Participant Rules.....	61
152	3.7.3.2 Session Authority Rules.....	62
153	3.8 Name Identifier Mapping Protocol.....	63
154	3.8.1 Element <NameIDMappingRequest>.....	63
155	3.8.2 Element <NameIDMappingResponse>.....	64
156	3.8.3 Processing Rules.....	64
157	4 SAML Versioning.....	65
158	4.1 SAML Specification Set Version.....	65
159	4.1.1 Schema Version.....	65
160	4.1.2 SAML Assertion Version.....	65
161	4.1.3 SAML Protocol Version.....	66
162	4.1.3.1 Request Version.....	66

163	4.1.3.2 Response Version.....	66
164	4.1.3.3 Permissible Version Combinations.....	67
165	4.2 SAML Namespace Version.....	67
166	4.2.1 Schema Evolution.....	67
167	5 SAML and XML Signature Syntax and Processing.....	68
168	5.1 Signing Assertions.....	68
169	5.2 Request/Response Signing.....	68
170	5.3 Signature Inheritance.....	68
171	5.4 XML Signature Profile.....	69
172	5.4.1 Signing Formats and Algorithms.....	69
173	5.4.2 References.....	69
174	5.4.3 Canonicalization Method.....	69
175	5.4.4 Transforms.....	69
176	5.4.5 KeyInfo.....	70
177	5.4.6 Example.....	70
178	6 SAML and XML Encryption Syntax and Processing.....	73
179	6.1 General Considerations.....	73
180	6.2 Combining Signatures and Encryption.....	73
181	7 SAML Extensibility.....	74
182	7.1 Schema Extension.....	74
183	7.1.1 Assertion Schema Extension.....	74
184	7.1.2 Protocol Schema Extension.....	74
185	7.2 Schema Wildcard Extension Points.....	75
186	7.2.1 Assertion Extension Points.....	75
187	7.2.2 Protocol Extension Points.....	75
188	7.3 Identifier Extension.....	75
189	8 SAML-Defined Identifiers.....	76
190	8.1 Action Namespace Identifiers.....	76
191	8.1.1 Read/Write/Execute/Delete/Control.....	76
192	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	76
193	8.1.3 Get/Head/Put/Post.....	77
194	8.1.4 UNIX File Permissions.....	77
195	8.2 Attribute Name Format Identifiers.....	77
196	8.2.1 Unspecified.....	77
197	8.2.2 URI Reference.....	78
198	8.2.3 Basic.....	78
199	8.3 Name Identifier Format Identifiers.....	78
200	8.3.1 Unspecified.....	78
201	8.3.2 Email Address.....	78
202	8.3.3 X.509 Subject Name.....	78
203	8.3.4 Windows Domain Qualified Name.....	78
204	8.3.5 Kerberos Principal Name.....	79
205	8.3.6 Entity Identifier.....	79
206	8.3.7 Persistent Identifier.....	79
207	8.3.8 Transient Identifier.....	80
208	8.4 Consent Identifiers.....	80
209	8.4.1 Unspecified.....	80
210	8.4.2 Obtained.....	80
211	8.4.3 Prior.....	80
212	8.4.4 Implicit.....	81
213	8.4.5 Explicit.....	81

214	8.4.6 Unavailable.....	81
215	8.4.7 Inapplicable.....	81
216	9 References.....	82
217	9.1 Normative References.....	82
218	9.2 Non-Normative References.....	82
219	Appendix A. Acknowledgments.....	84
220	Appendix B. Notices.....	86
221		



# 1 Introduction

The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This specification defines both the structure of SAML assertions, and an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML [XML] and use XML namespaces [XMLNS]. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. The SAML bindings specification [SAMLBind] provides frameworks for the embedding and transport of SAML protocol messages. The SAML profiles specification [SAMLProf] provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

For additional explanation of SAML terms and concepts, refer to the SAML technical overview [SAMLTechOvw] and the SAML glossary [SAMLGloss]. Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMPLP-XSD] are also available. The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

The following sections describe how to understand the rest of this specification.

## 1.1 Notation

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC 2119].

Listings of SAML schemas appear like this.

Example code listings appear like this.

**Note:** Notes like this are sometimes used to highlight non-normative commentary.

This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In cases of disagreement between the SAML schema documents and schema listings in this specification, the schema documents take precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace, defined in a schema [SAML-XSD]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace, defined in a schema [SAMPLP-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].

Prefix	XML Namespace	Comments
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

255 This specification uses the following typographical conventions in text: <SAML*Element*>,  
 256 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

## 257 1.2 Schema Organization and Namespaces

258 The SAML assertion structures are defined in a schema [SAML-XSD] associated with the following XML  
 259 namespace:

260 urn:oasis:names:tc:SAML:2.0:assertion

261 The SAML request-response protocol structures are defined in a schema [SAML-P-XSD] associated with  
 262 the following XML namespace:

263 urn:oasis:names:tc:SAML:2.0:protocol

264 The assertion schema is imported into the protocol schema. See Section 4.2 for information on SAML  
 265 namespace versioning.

266 Also imported into both schemas is the schema for XML Signature [XMLSig], which is associated with the  
 267 following XML namespace:

268 http://www.w3.org/2000/09/xmldsig#

269 Finally, the schema for XML Encryption [XMLEnc] is imported into the assertion schema and is associated  
 270 with the following XML namespace:

271 http://www.w3.org/2001/04/xmlenc#

## 272 1.3 Common Data Types

273 The following sections define how to use and interpret common data types that appear throughout the  
 274 SAML schemas.

### 275 1.3.1 String Values

276 All SAML string values have the type **xs:string**, which is built in to the W3C XML Schema Datatypes  
 277 specification [Schema2]. Unless otherwise noted in this specification or particular profiles, all strings in  
 278 SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the  
 279 XML Recommendation [XML] Section 2.3).

280 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that  
 281 have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact  
 282 binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-  
 283 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific

formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

If an implementation is comparing values that are represented using different character encodings, the implementation **MUST** use a comparison method that returns the same result as converting both values to the Unicode character encoding, Normalization Form C [UNICODE-C], and then performing an exact binary comparison. This requirement is intended to conform to the W3C Character Model for the World Wide Web [W3C-CharMod], and in particular the rules for Unicode-normalized Text.

Applications that compare data received in SAML documents to data from external sources **MUST** take into account the normalization rules specified for XML. Text contained within elements is normalized so that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the XML Recommendation [XML] Section 2.11. XML attribute values defined as strings (or types derived from strings) are normalized as described in [XML] Section 3.3.3. All whitespace characters are replaced with blanks (ASCII code 32<sub>Decimal</sub>).

The SAML specification does not define collation or sorting order for XML attribute values or element content. SAML implementations **MUST NOT** depend on specific sorting orders for values, because these can differ depending on the locale settings of the hosts involved.

### 1.3.2 URI Values

All SAML URI reference values have the type **xs:anyURI**, which is built in to the W3C XML Schema Datatypes specification [Schema2].

Unless otherwise indicated in this specification, all URI reference values used within SAML-defined elements or attributes **MUST** consist of at least one non-whitespace character, and are **REQUIRED** to be absolute [RFC 2396].

Note that the SAML specification makes extensive use of URI references as identifiers, such as status codes, format types, attribute and system entity names, etc. In such cases, it is essential that the values be both unique and consistent, such that the same URI is never used at different times to represent different underlying information.

### 1.3.3 Time Values

All SAML time values have the type **xs:dateTime**, which is built in to the W3C XML Schema Datatypes specification [Schema2], and **MUST** be expressed in UTC form, with no time zone component.

SAML system entities **SHOULD NOT** rely on time resolution finer than milliseconds. Implementations **MUST NOT** generate time instants that specify leap seconds.

### 1.3.4 ID and ID Reference Values

The **xs:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type **xs:ID** in this specification **MUST** satisfy the following properties in addition to those imposed by the definition of the **xs:ID** type itself:

- Any party that assigns an identifier **MUST** ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object.
- Where a data object declares that it has a particular identifier, there **MUST** be exactly one such declaration.

The mechanism by which a SAML system entity ensures that the identifier is unique is left to the implementation. In the case that a random or pseudorandom technique is employed, the probability of two randomly chosen identifiers being identical **MUST** be less than or equal to  $2^{-128}$  and **SHOULD** be less than or equal to  $2^{-160}$ . This requirement **MAY** be met by encoding a randomly chosen value between 128 and

327 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom  
328 generator **MUST** be seeded with unique material in order to ensure the desired uniqueness properties  
329 between different systems.

330 The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs:IDREF**  
331 cannot be used for this purpose. In SAML, the element referred to by a SAML identifier reference might  
332 actually be defined in a document separate from that in which the identifier reference is used. Using  
333 **xs:IDREF** would violate the requirement that its value match the value of an ID attribute on some element  
334 in the same XML document.

335 **Note:** It is anticipated that the World Wide Web Consortium will standardize a global  
336 attribute for holding ID-typed values, called `xml:id` [XML-ID]. The Security Services  
337 Technical Committee plans to move away from SAML-specific ID attributes to this style of  
338 assigning unique identifiers as soon as practicable after the `xml:id` attribute is  
339 standardized.

## 2 SAML Assertions

An assertion is a package of information that supplies zero or more statements made by a **SAML authority**; SAML authorities are sometimes referred to as **asserting parties** in discussions of assertion generation and exchange, and system entities that use received assertions are known as **relying parties**. (Note that these terms are different from **requester** and **responder**, which are reserved for discussions of SAML protocol message exchange.)

SAML assertions are usually made about a **subject**, represented by the <Subject> element. However, the <Subject> element is optional, and other specifications and profiles may utilize the SAML assertion structure to make similar statements without specifying a subject, or possibly specifying the subject in an alternate way. Typically there are a number of **service providers** that can make use of assertions about a subject in order to control access and provide customized service, and accordingly they become the relying parties of an asserting party called an **identity provider**.

This SAML specification defines three different kinds of assertion statements that can be created by a SAML authority. All SAML-defined statements are associated with a subject. The three kinds of statement defined in this specification are:

- **Authentication:** The assertion subject was authenticated by a particular means at a particular time.
- **Attribute:** The assertion subject is associated with the supplied attributes.
- **Authorization Decision:** A request to allow the assertion subject to access the specified resource has been granted or denied.

The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, attribute, authorization decision, or user-defined statements containing the specifics.

As described in Section 7, extensions are permitted by the SAML assertion schema, allowing user-defined extensions to assertions and statements, as well as allowing the definition of new kinds of assertions and statements.

The SAML technical overview [SAMLTechOvw] and glossary [SAMLGloss] provide more detailed explanation of SAML terms and concepts.

### 2.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the assertion schema:

```
<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
  <import namespace="http://www.w3.org/2001/04/xmlenc#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
20021210/xenc-schema.xsd"/>
  <annotation>
    <documentation>
      Document identifier: saml-schema-assertion-2.0
```

```

388      Location: http://docs.oasis-open.org/security/saml/v2.0/
389      Revision history:
390      V1.0 (November, 2002):
391      Initial Standard Schema.
392      V1.1 (September, 2003):
393      Updates within the same V1.0 namespace.
394      V2.0 (March, 2005):
395      New assertion schema for SAML V2.0 namespace.
396    </documentation>
397  </annotation>
398  ...
399 </schema>

```

## 2.2 Name Identifiers

The following sections define the SAML constructs that contain descriptive identifiers for subjects and the issuers of assertions and protocol messages.

There are a number of circumstances in SAML in which it is useful for two system entities to communicate regarding a third party; for example, the SAML authentication request protocol enables third-party authentication of a subject. Thus, it is useful to establish a means by which parties may be associated with identifiers that are meaningful to each of the parties. In some cases, it will be necessary to limit the scope within which an identifier is used to a small set of system entities (to preserve the privacy of a subject, for example). Similar identifiers may also be used to refer to the issuer of a SAML protocol message or assertion.

It is possible that two or more system entities may use the same name identifier value when referring to different identities. Thus, each entity may have a different understanding of that same name. SAML provides **name qualifiers** to disambiguate a name identifier by effectively placing it in a federated **namespace** related to the name qualifiers. SAML V2.0 allows an identifier to be qualified in terms of both an asserting party and a particular relying party or affiliation, allowing identifiers to exhibit pair-wise semantics, when required.

Name identifiers may also be encrypted to further improve their privacy-preserving characteristics, particularly in cases where the identifier may be transmitted via an intermediary.

**Note:** To avoid use of relatively advanced XML schema constructs (among other reasons), the various types of identifier elements do not share a common type hierarchy.

### 2.2.1 Element <BaseID>

The <BaseID> element is an extension point that allows applications to add new kinds of identifiers. Its **BaseIDAbstractType** complex type is abstract and is thus usable only as the base of a derived type. It includes the following attributes for use by extended identifier representations:

#### NameQualifier [Optional]

The security or administrative domain that qualifies the identifier. This attribute provides a means to federate identifiers from disparate user stores without collision.

#### SPNameQualifier [Optional]

Further qualifies an identifier with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate identifiers on the basis of the relying party or parties.

The **NameQualifier** and **SPNameQualifier** attributes SHOULD be omitted unless the identifier's type definition explicitly defines their use and semantics.



The following schema fragment defines the <BaseID> element and its **BaseIDAbstractType** complex type:

```
<attributeGroup name="IDNameQualifiers">
  <attribute name="NameQualifier" type="string" use="optional"/>
  <attribute name="SPNameQualifier" type="string" use="optional"/>
</attributeGroup>
<element name="BaseID" type="saml:BaseIDAbstractType"/>
<complexType name="BaseIDAbstractType" abstract="true">
  <attributeGroup ref="saml:IDNameQualifiers"/>
</complexType>
```

## 2.2.2 Complex Type NameIDType

The **NameIDType** complex type is used when an element serves to represent an entity by a string-valued name. It is a more restricted form of identifier than the <BaseID> element and is the type underlying both the <NameID> and <Issuer> elements. In addition to the string content containing the actual identifier, it provides the following optional attributes:

### NameQualifier [Optional]

The security or administrative domain that qualifies the name. This attribute provides a means to federate names from disparate user stores without collision.

### SPNameQualifier [Optional]

Further qualifies a name with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate names on the basis of the relying party or parties.

### Format [Optional]

A URI reference representing the classification of string-based identifier information. See Section 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute and their associated descriptions and processing rules. Unless otherwise specified by an element based on this type, if no `Format` value is provided, then the value `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in effect.

When a `Format` value other than one specified in Section 8.3 is used, the content of an element of this type is to be interpreted according to the definition of that format as provided outside of this specification. If not otherwise indicated by the definition of the format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties are implementation-specific.

### SPProvidedID [Optional]

A name identifier established by a service provider or affiliation of providers for the entity, if different from the primary name identifier given in the content of the element. This attribute provides a means of integrating the use of SAML with existing identifiers already in use by a service provider. For example, an existing identifier can be "attached" to the entity using the Name Identifier Management protocol defined in Section 3.6.

Additional rules for the content of (or the omission of) these attributes can be defined by elements that make use of this type, and by specific `Format` definitions. The `NameQualifier` and `SPNameQualifier` attributes SHOULD be omitted unless the element or format explicitly defines their use and semantics.

The following schema fragment defines the **NameIDType** complex type:

```
<complexType name="NameIDType">
  <simpleContent>
```

```

479     <extension base="string">
480       <attributeGroup ref="saml:IDNameQualifiers"/>
481       <attribute name="Format" type="anyURI" use="optional"/>
482       <attribute name="SPProvidedID" type="string" use="optional"/>
483     </extension>
484   </simpleContent>
485 </complexType>

```

### 2.2.3 Element <NameID>

The <NameID> element is of type **NameIDType** (see Section 2.2.2), and is used in various SAML assertion constructs such as the <Subject> and <SubjectConfirmation> elements, and in various protocol messages (see Section 3).

The following schema fragment defines the <NameID> element:

```

491 <element name="NameID" type="saml:NameIDType"/>

```

### 2.2.4 Element <EncryptedID>

The <EncryptedID> element is of type **EncryptedElementType**, and carries the content of an unencrypted identifier element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedID> element contains the following elements:

<xenc:EncryptedData> [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The *Type* attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of **NameIDType** or **AssertionType**, or a type that is derived from **BaseIDAbstractType**, **NameIDType**, or **AssertionType**.

<xenc:EncryptedKey> [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a *Recipient* attribute that specifies the entity for whom the key has been encrypted. The value of the *Recipient* attribute SHOULD be the URI identifier of a SAML system entity, as defined by Section 8.3.6.

Encrypted identifiers are intended as a privacy protection mechanism when the plain-text value passes through an intermediary. As such, the ciphertext MUST be unique to any given encryption operation. For more on such issues, see [XMLEnc] Section 6.3.

Note that an entire assertion can be encrypted into this element and used as an identifier. In such a case, the <Subject> element of the encrypted assertion supplies the "identifier" of the subject of the enclosing assertion. Note also that if the identifying assertion is invalid, then so is the enclosing assertion.

The following schema fragment defines the <EncryptedID> element and its **EncryptedElementType** complex type:

```

515 <complexType name="EncryptedElementType">
516   <sequence>
517     <element ref="xenc:EncryptedData"/>
518     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
519   </sequence>
520 </complexType>
521 <element name="EncryptedID" type="saml:EncryptedElementType"/>

```

## 2.2.5 Element <Issuer>

The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name, but permits various pieces of descriptive data (see Section 2.2.2).

Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see Section 8.3.6).

The following schema fragment defines the <Issuer> element:

```
<element name="Issuer" type="saml:NameIDType"/>
```

## 2.3 Assertions

The following sections define the SAML constructs that either contain assertion information or provide a means to refer to an existing assertion.

### 2.3.1 Element <AssertionIDRef>

The <AssertionIDRef> element makes a reference to a SAML assertion by its unique identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is not specified as part of the reference. See Section 3.3.1 for a protocol element that uses such a reference to ask for the corresponding assertion.

The following schema fragment defines the <AssertionIDRef> element:

```
<element name="AssertionIDRef" type="NCName"/>
```

### 2.3.2 Element <AssertionURIRef>

The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. The URI reference MAY be used to retrieve the corresponding assertion in a manner specific to the URI reference. See Section 3.7 of the Bindings specification [SAMLBind] for information on how this element is used in a protocol binding to accomplish this.

The following schema fragment defines the <AssertionURIRef> element:

```
<element name="AssertionURIRef" type="anyURI"/>
```

### 2.3.3 Element <Assertion>

The <Assertion> element is of the **AssertionType** complex type. This type specifies the basic information that is common to all assertions, including the following elements and attributes:

**Version** [Required]

The version of this assertion. The identifier for the version of SAML defined in this specification is "2.0". SAML versioning is discussed in Section 4.

**ID** [Required]

The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness.

**IssueInstant** [Required]

The time instant of issue in UTC, as described in Section 1.3.3.

558 <Issuer> [Required]  
 559 The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be unambiguous  
 560 to the intended relying parties.

561 This specification defines no particular relationship between the entity represented by this element  
 562 and the signer of the assertion (if any). Any such requirements imposed by a relying party that  
 563 consumes the assertion or by specific profiles are application-specific.

564 <ds:Signature> [Optional]  
 565 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as  
 566 described below and in Section 5.

567 <Subject> [Optional]  
 568 The subject of the statement(s) in the assertion.

569 <Conditions> [Optional]  
 570 Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion.  
 571 See Section 2.5 for additional information on how to evaluate conditions.

572 <Advice> [Optional]  
 573 Additional information related to the assertion that assists processing in certain situations but which  
 574 MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

575 Zero or more of the following statement elements:

576 <Statement>  
 577 A statement of a type defined in an extension schema. An `xsi:type` attribute MUST be used to  
 578 indicate the actual statement type.

579 <AuthnStatement>  
 580 An authentication statement.

581 <AuthzDecisionStatement>  
 582 An authorization decision statement.

583 <AttributeStatement>  
 584 An attribute statement.

585 An assertion with no statements MUST contain a <Subject> element. Such an assertion identifies a  
 586 principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further  
 587 information associated with that principal.

588 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If  
 589 <Subject> is omitted, then the statements in the assertion apply to a subject or subjects identified in an  
 590 application- or profile-specific manner. SAML itself defines no such statements, and an assertion without a  
 591 subject has no defined meaning in this specification.

592 Depending on the requirements of particular protocols or profiles, the issuer of a SAML assertion may  
 593 often need to be authenticated, and integrity protection may often be required. Authentication and  
 594 message integrity MAY be provided by mechanisms provided by a protocol binding in use during the  
 595 delivery of an assertion (see [SAMLBind]). The SAML assertion MAY be signed, which provides both  
 596 authentication of the issuer and integrity protection.

597 If such a signature is used, then the <ds:Signature> element MUST be present, and a relying party  
 598 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in  
 599 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the  
 600 assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity and  
 601 appropriateness of the issuer and may continue to process the assertion in accordance with this

specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-specific rules, and so on).

Note that whether signed or unsigned, the inclusion of multiple statements within a single assertion is semantically equivalent to a set of assertions containing those statements individually (provided the subject, conditions, etc. are also the same).

The following schema fragment defines the `<Assertion>` element and its **AssertionType** complex type:

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
  <sequence>
    <element ref="saml:Issuer"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="saml:Subject" minOccurs="0"/>
    <element ref="saml:Conditions" minOccurs="0"/>
    <element ref="saml:Advice" minOccurs="0"/>
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="saml:Statement"/>
      <element ref="saml:AuthnStatement"/>
      <element ref="saml:AuthzDecisionStatement"/>
      <element ref="saml:AttributeStatement"/>
    </choice>
  </sequence>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

### 2.3.4 Element `<EncryptedAssertion>`

The `<EncryptedAssertion>` element represents an assertion in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAssertion>` element contains the following elements:

`<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of or derived from **AssertionType**.

`<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by Section 8.3.6.

Encrypted assertions are intended as a confidentiality protection mechanism when the plain-text value passes through an intermediary.

The following schema fragment defines the `<EncryptedAssertion>` element:

```
<element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
```

## 2.4 Subjects

This section defines the SAML constructs used to describe the subject of an assertion.

## 2.4.1 Element <Subject>

The optional <Subject> element specifies the principal that is the subject of all of the (zero or more) statements in the assertion. It contains an identifier, a series of one or more subject confirmations, or both:

<BaseID>, <NameID>, or <EncryptedID> [Optional]

Identifies the subject.

<SubjectConfirmation> [Zero or More]

Information that allows the subject to be confirmed. If more than one subject confirmation is provided, then satisfying any one of them is sufficient to confirm the subject for the purpose of applying the assertion.

A <Subject> element can contain both an identifier and zero or more subject confirmations which a relying party can verify when processing an assertion. If any one of the included subject confirmations are verified, the relying party MAY treat the entity presenting the assertion as one that the asserting party has associated with the principal identified in the name identifier and associated with the statements in the assertion. This attesting entity and the actual subject may or may not be the same entity.

If there are no subject confirmations included, then any relationship between the presenter of the assertion and the actual subject is unspecified.

A <Subject> element SHOULD NOT identify more than one principal.

The following schema fragment defines the <Subject> element and its **SubjectType** complex type:

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
  <choice>
    <sequence>
      <choice>
        <element ref="saml:BaseID"/>
        <element ref="saml:NameID"/>
        <element ref="saml:EncryptedID"/>
      </choice>
      <element ref="saml:SubjectConfirmation" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
  </choice>
</complexType>
```

### 2.4.1.1 Element <SubjectConfirmation>

The <SubjectConfirmation> element provides the means for a relying party to verify the correspondence of the subject of the assertion with the party with whom the relying party is communicating. It contains the following attributes and elements:

Method [Required]

A URI reference that identifies a protocol or mechanism to be used to confirm the subject. URI references identifying SAML-defined confirmation methods are currently defined in the SAML profiles specification [SAMLProf]. Additional methods MAY be added by defining new URIs and profiles or by private agreement.

<BaseID>, <NameID>, or <EncryptedID> [Optional]

Identifies the entity expected to satisfy the enclosing subject confirmation requirements.



692 <SubjectConfirmationData> [Optional]

693 Additional confirmation information to be used by a specific confirmation method. For example, typical  
 694 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax  
 695 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section  
 696 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,  
 697 attributes, or content that may appear in the <SubjectConfirmationData> element.

698 The following schema fragment defines the <SubjectConfirmation> element and its  
 699 **SubjectConfirmationType** complex type:

```

700 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
701 <complexType name="SubjectConfirmationType">
702   <sequence>
703     <choice minOccurs="0">
704       <element ref="saml:BaseID"/>
705       <element ref="saml:NameID"/>
706       <element ref="saml:EncryptedID"/>
707     </choice>
708     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
709   </sequence>
710   <attribute name="Method" type="anyURI" use="required"/>
711 </complexType>

```

#### 712 2.4.1.2 Element <SubjectConfirmationData>

713 The <SubjectConfirmationData> element has the **SubjectConfirmationDataType** complex type. It  
 714 specifies additional data that allows the subject to be confirmed or constrains the circumstances under  
 715 which the act of subject confirmation can take place. Subject confirmation takes place when a relying  
 716 party seeks to verify the relationship between an entity presenting the assertion (that is, the attesting  
 717 entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply  
 718 to any method:

719 NotBefore [Optional]

720 A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as  
 721 described in Section 1.3.3.

722 NotOnOrAfter [Optional]

723 A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as  
 724 described in Section 1.3.3.

725 Recipient [Optional]

726 A URI specifying the entity or location to which an attesting entity can present the assertion. For  
 727 example, this attribute might indicate that the assertion must be delivered to a particular network  
 728 endpoint in order to prevent an intermediary from redirecting it someplace else.

729 InResponseTo [Optional]

730 The ID of a SAML protocol message in response to which an attesting entity can present the  
 731 assertion. For example, this attribute might be used to correlate the assertion to a SAML request that  
 732 resulted in its presentation.

733 Address [Optional]

734 The network address/location from which an attesting entity can present the assertion. For example,  
 735 this attribute might be used to bind the assertion to particular client addresses to prevent an attacker  
 736 from easily stealing and presenting the assertion from another location. IPv4 addresses SHOULD be  
 737 represented in the usual dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses SHOULD be  
 738 represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g.,  
 739 "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

#### 740 Arbitrary attributes

741 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary namespace-  
 742 qualified XML attributes to be added to `<SubjectConfirmationData>` constructs without the need  
 743 for an explicit schema extension. This allows additional fields to be added as needed to supply  
 744 additional confirmation-related information. SAML extensions MUST NOT add local (non-namespace-  
 745 qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the  
 746 **SubjectConfirmationDataType** complex type or a derivation of it; such attributes are reserved for  
 747 future maintenance and enhancement of SAML itself.

#### 748 Arbitrary elements

749 This complex type uses an `<xs:any>` extension point to allow arbitrary XML elements to be added to  
 750 `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This  
 751 allows additional elements to be added as needed to supply additional confirmation-related  
 752 information.

753 Particular confirmation methods and profiles that make use of those methods MAY require the use of one  
 754 or more of the attributes defined within this complex type. For examples of how these attributes (and  
 755 subject confirmation in general) can be used, see the Profiles specification [SAMLProf].

756 Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present,  
 757 SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's  
 758 `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore`  
 759 MUST be less than (earlier than) the value for `NotOnOrAfter`.

760 The following schema fragment defines the `<SubjectConfirmationData>` element and its  
 761 **SubjectConfirmationDataType** complex type:

```

762 <element name="SubjectConfirmationData"
763   type="saml:SubjectConfirmationDataType"/>
764 <complexType name="SubjectConfirmationDataType" mixed="true">
765   <complexContent>
766     <restriction base="anyType">
767       <sequence>
768         <any namespace="##any" processContents="lax" minOccurs="0"
769 maxOccurs="unbounded"/>
770       </sequence>
771       <attribute name="NotBefore" type="dateTime" use="optional"/>
772       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
773       <attribute name="Recipient" type="anyURI" use="optional"/>
774       <attribute name="InResponseTo" type="NCName" use="optional"/>
775       <attribute name="Address" type="string" use="optional"/>
776       <anyAttribute namespace="##other" processContents="lax"/>
777     </restriction>
778   </complexContent>
779 </complexType>

```

#### 780 2.4.1.3 Complex Type KeyInfoConfirmationDataType

781 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`  
 782 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in  
 783 some way to authenticate an attesting entity. The particular confirmation method MUST define the exact  
 784 mechanism by which the confirmation data can be used. The optional attributes defined by the  
 785 **SubjectConfirmationDataType** complex type MAY also appear.

786 This complex type, or a type derived from it, SHOULD be used by any confirmation method that defines its  
 787 confirmation data in terms of the `<ds:KeyInfo>` element.

Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element MUST identify a single cryptographic key. Multiple keys MAY be identified with separate `<ds:KeyInfo>` elements, such as when a principal uses different keys to confirm itself to different relying parties.

The following schema fragment defines the **KeyInfoConfirmationDataType** complex type:

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">
  <complexContent>
    <restriction base="saml:SubjectConfirmationDataType">
      <sequence>
        <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
```

#### 2.4.1.4 Example of a Key-Confirmed <Subject>

To illustrate the way in which the various elements and types fit together, below is an example of a `<Subject>` element containing a name identifier and a subject confirmation based on proof of possession of a key. Note the use of the **KeyInfoConfirmationDataType** to identify the confirmation data syntax as being a `<ds:KeyInfo>` element:

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
    scott@example.org
  </NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
    <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
      <ds:KeyInfo>
        <ds:KeyName>Scott's Key</ds:KeyName>
      </ds:KeyInfo>
    </SubjectConfirmationData>
  </SubjectConfirmation>
</Subject>
```

## 2.5 Conditions

This section defines the SAML constructs that place constraints on the acceptable use of SAML assertions.

### 2.5.1 Element <Conditions>

The `<Conditions>` element MAY contain the following elements and attributes:

**NotBefore** [Optional]

Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as described in Section 1.3.3.

**NotOnOrAfter** [Optional]

Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as described in Section 1.3.3.

**<Condition>** [Any Number]

A condition of a type defined in an extension schema. An `xsi:type` attribute MUST be used to indicate the actual condition type.

**<AudienceRestriction>** [Any Number]

Specifies that the assertion is addressed to a particular audience.

<OneTimeUse> [Optional]

Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future use. Although the schema permits multiple occurrences, there MUST be at most one instance of this element.

<ProxyRestriction> [Optional]

Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act as asserting parties themselves and issue assertions of their own on the basis of the information contained in the original assertion. Although the schema permits multiple occurrences, there MUST be at most one instance of this element.

Because the use of the `xsi:type` attribute would permit an assertion to contain more than one instance of a SAML-defined subtype of **ConditionsType** (such as **OneTimeUseType**), the schema does not explicitly limit the number of times particular conditions may be included. A particular type of condition MAY define limits on such use, as shown above.

The following schema fragment defines the <Conditions> element and its **ConditionsType** complex type:

```
<element name="Conditions" type="saml:ConditionsType"/>
<complexType name="ConditionsType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="saml:Condition"/>
    <element ref="saml:AudienceRestriction"/>
    <element ref="saml:OneTimeUse"/>
    <element ref="saml:ProxyRestriction"/>
  </choice>
  <attribute name="NotBefore" type="dateTime" use="optional"/>
  <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
```

### 2.5.1.1 General Processing Rules

If an assertion contains a <Conditions> element, then the validity of the assertion is dependent on the sub-elements and attributes provided, using the following rules in the order shown below.

Note that an assertion that has condition validity status **Valid** may nonetheless be untrustworthy or invalid for reasons such as not being well-formed or schema-valid, not being issued by a trustworthy SAML authority, or not being authenticated by a trustworthy means.

Also note that some conditions may not directly impact the validity of the containing assertion (they always evaluate to **Valid**), but may restrict the behavior of relying parties with respect to the use of the assertion.

1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is considered to be **Valid** with respect to condition processing.
2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the assertion is considered to be **Invalid**.
3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, or if an element is encountered that is not understood, then the validity of the assertion cannot be determined and is considered to be **Indeterminate**.
4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the assertion is considered to be **Valid** with respect to condition processing.

The first rule that applies terminates condition processing; thus a determination that an assertion is **Invalid** takes precedence over that of **Indeterminate**.

An assertion that is determined to be **Invalid** or **Indeterminate** MUST be rejected by a relying party (within whatever context or profile it was being processed), just as if the assertion were malformed or otherwise unusable.

### 2.5.1.2 Attributes **NotBefore** and **NotOnOrAfter**

The **NotBefore** and **NotOnOrAfter** attributes specify time limits on the validity of the assertion within the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be correct or accurate throughout the validity period.

The **NotBefore** attribute specifies the time instant at which the validity interval begins. The **NotOnOrAfter** attribute specifies the time instant at which the validity interval has ended.

If the value for either **NotBefore** or **NotOnOrAfter** is omitted, then it is considered unspecified. If the **NotBefore** attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then the assertion is **Valid** with respect to conditions at any time before the time instant specified by the **NotOnOrAfter** attribute. If the **NotOnOrAfter** attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant specified by the **NotBefore** attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any time.

If both attributes are present, the value for **NotBefore** MUST be less than (earlier than) the value for **NotOnOrAfter**.

### 2.5.1.3 Element **<Condition>**

The **<Condition>** element serves as an extension point for new conditions. Its **ConditionAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the **<Condition>** element and its **ConditionAbstractType** complex type:

```
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
```

### 2.5.1.4 Elements **<AudienceRestriction>** and **<Audience>**

The **<AudienceRestriction>** element specifies that the assertion is addressed to one or more specific audiences identified by **<Audience>** elements. Although a SAML relying party that is outside the audiences specified is capable of drawing conclusions from an assertion, the SAML asserting party explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the following element:

**<Audience>**

A URI reference that identifies an intended audience. The URI reference MAY identify a document that describes the terms and conditions of audience membership. It MAY also contain the unique identifier URI from a SAML name identifier that describes a system entity (see Section 8.3.6).

The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of one or more of the audiences specified.

The SAML asserting party cannot prevent a party to whom the assertion is disclosed from taking action on the basis of the information provided. However, the **<AudienceRestriction>** element allows the SAML asserting party to state explicitly that no warranty is provided to such a party in a machine- and human-readable form. While there can be no guarantee that a court would uphold such a warranty exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably improved.

Note that multiple `<AudienceRestriction>` elements MAY be included in a single assertion, and each MUST be evaluated independently. The effect of this requirement and the preceding definition is that within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a conjunction (an "AND").

The following schema fragment defines the `<AudienceRestriction>` element and its **AudienceRestrictionType** complex type:

```
<element name="AudienceRestriction"
  type="saml:AudienceRestrictionType"/>
<complexType name="AudienceRestrictionType">
  <complexContent>
    <extension base="saml:ConditionAbstractType">
      <sequence>
        <element ref="saml:Audience" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

### 2.5.1.5 Element `<OneTimeUse>`

In general, relying parties may choose to retain assertions, or the information they contain in some other form, for reuse. The `<OneTimeUse>` condition element allows an authority to indicate that the information in the assertion is likely to change very soon and fresh information should be obtained for each use. An example would be an assertion containing an `<AuthzDecisionStatement>` which was the result of a policy which specified access control which was a function of the time of day.

If system clocks in a distributed environment could be precisely synchronized, then this requirement could be met by careful use of the validity interval. However, since some clock skew between systems will always be present and will be combined with possible transmission delays, there is no convenient way for the issuer to appropriately limit the lifetime of an assertion without running a substantial risk that it will already have expired before it arrives.

The `<OneTimeUse>` element indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be retained for future use. Relying parties are always free to request a fresh assertion for every use. However, implementations that choose to retain assertions for future use MUST observe the `<OneTimeUse>` element. This condition is independent from the `NotBefore` and `NotOnOrAfter` condition information.

To support the single use constraint, a relying party should maintain a cache of the assertions it has processed containing such a condition. Whenever an assertion with this condition is processed, the cache should be checked to ensure that the same assertion has not been previously received and processed by the relying party.

A SAML authority MUST NOT include more than one `<OneTimeUse>` element within a `<Conditions>` element of an assertion.

For the purposes of determining the validity of the `<Conditions>` element, the `<OneTimeUse>` is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the `<OneTimeUse>` element and its **OneTimeUseType** complex type:

```
<element name="OneTimeUse" type="saml:OneTimeUseType"/>
<complexType name="OneTimeUseType">
  <complexContent>
    <extension base="saml:ConditionAbstractType"/>
  </complexContent>
</complexType>
```



### 2.5.1.6 Element <ProxyRestriction>

Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting parties and issue subsequent assertions of their own on the basis of the information contained in the original assertion. A relying party acting as an asserting party MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

The <ProxyRestriction> element contains the following elements and attributes:

**Count** [Optional]

Specifies the maximum number of indirections that the asserting party permits to exist between this assertion and an assertion which has ultimately been issued on the basis of it.

**<Audience>** [Zero or More]

Specifies the set of audiences to whom the asserting party permits new assertions to be issued on the basis of this assertion.

A **Count** value of zero indicates that a relying party MUST NOT issue an assertion to another relying party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves contain a <ProxyRestriction> element with a **Count** value of at most one less than this value.

If no <Audience> elements are specified, then no audience restrictions are imposed on the relying parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves contain an <AudienceRestriction> element with at least one of the <Audience> elements present in the previous <ProxyRestriction> element, and no <Audience> elements present that were not in the previous <ProxyRestriction> element.

A SAML authority MUST NOT include more than one <ProxyRestriction> element within a <Conditions> element of an assertion.

For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction> condition is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the <ProxyRestriction> element and its **ProxyRestrictionType** complex type:

```
<element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
<complexType name="ProxyRestrictionType">
  <complexContent>
    <extension base="saml:ConditionAbstractType">
      <sequence>
        <element ref="saml:Audience" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="Count" type="nonNegativeInteger" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

## 2.6 Advice

This section defines the SAML constructs that contain additional information about an assertion that an asserting party wishes to provide to a relying party.

## 2.6.1 Element <Advice>

The <Advice> element contains any additional information that the SAML authority wishes to provide. This information MAY be ignored by applications without affecting either the semantics or the validity of the assertion.

The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>, <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in other non-SAML namespaces.

Following are some potential uses of the <Advice> element:

- Include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions).
- State a proof of the assertion claims.
- Specify the timing and distribution points for updates to the assertion.

The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
<element name="Advice" type="saml:AdviceType"/>
<complexType name="AdviceType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="saml:AssertionIDRef"/>
    <element ref="saml:AssertionURIRef"/>
    <element ref="saml:Assertion"/>
    <element ref="saml:EncryptedAssertion"/>
    <any namespace="##other" processContents="lax"/>
  </choice>
</complexType>
```

## 2.7 Statements

The following sections define the SAML constructs that contain statement information.

### 2.7.1 Element <Statement>

The <Statement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. SAML itself derives its core statements from this extension point. Its **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the <Statement> element and its **StatementAbstractType** complex type:

```
<element name="Statement" type="saml:StatementAbstractType"/>
<complexType name="StatementAbstractType" abstract="true"/>
```

### 2.7.2 Element <AuthnStatement>

The <AuthnStatement> element describes a statement by the SAML authority asserting that the assertion subject was authenticated by a particular means at a particular time. Assertions containing <AuthnStatement> elements MUST contain a <Subject> element.

It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

**Note:** The <AuthorityBinding> element and its corresponding type were removed from <AuthnStatement> for V2.0 of SAML.

1055 **AuthnInstant** [Required]

1056 Specifies the time at which the authentication took place. The time value is encoded in UTC, as  
 1057 described in Section 1.3.3.

1058 **SessionIndex** [Optional]

1059 Specifies the index of a particular session between the principal identified by the subject and the  
 1060 authenticating authority.

1061 **SessionNotOnOrAfter** [Optional]

1062 Specifies a time instant at which the session between the principal identified by the subject and the  
 1063 SAML authority issuing this statement **MUST** be considered ended. The time value is encoded in  
 1064 UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a  
 1065 **NotOnOrAfter** condition attribute that may be present in the assertion.

1066 **<SubjectLocality>** [Optional]

1067 Specifies the DNS domain name and IP address for the system from which the assertion subject was  
 1068 apparently authenticated.

1069 **<AuthnContext>** [Required]

1070 The context used by the authenticating authority up to and including the authentication event that  
 1071 yielded this statement. Contains an authentication context class reference, an authentication context  
 1072 declaration or declaration reference, or both. See the Authentication Context specification  
 1073 [SAMLAuthnCxt] for a full description of authentication context information.

1074 In general, any string value **MAY** be used as a **SessionIndex** value. However, when privacy is a  
 1075 consideration, care must be taken to ensure that the **SessionIndex** value does not invalidate other  
 1076 privacy mechanisms. Accordingly, the value **SHOULD NOT** be usable to correlate activity by a principal  
 1077 across different session participants. Two solutions that achieve this goal are provided below and are  
 1078 **RECOMMENDED**:

- 1079 • Use small positive integers (or reoccurring constants in a list) for the **SessionIndex**. The SAML  
 1080 authority **SHOULD** choose the range of values such that the cardinality of any one integer will be  
 1081 sufficiently high to prevent a particular principal's actions from being correlated across multiple session  
 1082 participants. The SAML authority **SHOULD** choose values for **SessionIndex** randomly from within  
 1083 this range (except when required to ensure unique values for subsequent statements given to the  
 1084 same session participant but as part of a distinct session).
- 1085 • Use the enclosing assertion's **ID** value in the **SessionIndex**.

1086 The following schema fragment defines the **<AuthnStatement>** element and its **AuthnStatementType**  
 1087 complex type:

```

1088 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1089 <complexType name="AuthnStatementType">
1090   <complexContent>
1091     <extension base="saml:StatementAbstractType">
1092       <sequence>
1093         <element ref="saml:SubjectLocality" minOccurs="0"/>
1094         <element ref="saml:AuthnContext"/>
1095       </sequence>
1096       <attribute name="AuthnInstant" type="dateTime" use="required"/>
1097       <attribute name="SessionIndex" type="string" use="optional"/>
1098       <attribute name="SessionNotOnOrAfter" type="dateTime"
1099 use="optional"/>
1100     </extension>
1101   </complexContent>
1102 </complexType>
  
```

### 2.7.2.1 Element <SubjectLocality>

The <SubjectLocality> element specifies the DNS domain name and IP address for the system from which the assertion subject was authenticated. It has the following attributes:

Address [Optional]

The network address of the system from which the principal identified by the subject was authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

DNSName [Optional]

The DNS name of the system from which the principal identified by the subject was authenticated.

This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful information in some applications.

The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType** complex type:

```
<element name="SubjectLocality" type="saml:SubjectLocalityType"/>
<complexType name="SubjectLocalityType">
  <attribute name="Address" type="string" use="optional"/>
  <attribute name="DNSName" type="string" use="optional"/>
</complexType>
```

### 2.7.2.2 Element <AuthnContext>

The <AuthnContext> element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both. Its complex **AuthnContextType** has the following elements:

<AuthnContextClassRef> [Optional]

A URI reference identifying an authentication context class that describes the authentication context declaration that follows.

<AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

Either an authentication context declaration provided by value, or a URI reference that identifies such a declaration. The URI reference MAY directly resolve into an XML document containing the referenced declaration.

<AuthenticatingAuthority> [Zero or More]

Zero or more unique identifiers of authentication authorities that were involved in the authentication of the principal (not including the assertion issuer, who is presumed to have been involved without being explicitly named here).

See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication context information.

The following schema fragment defines the <AuthnContext> element and its **AuthnContextType** complex type:

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
  <sequence>
    <choice>
      <sequence>
        <element ref="saml:AuthnContextClassRef"/>
        <choice minOccurs="0">
```

```

1148         <element ref="saml:AuthnContextDecl"/>
1149         <element ref="saml:AuthnContextDeclRef"/>
1150     </choice>
1151 </sequence>
1152 <choice>
1153     <element ref="saml:AuthnContextDecl"/>
1154     <element ref="saml:AuthnContextDeclRef"/>
1155 </choice>
1156 </choice>
1157 <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1158 maxOccurs="unbounded"/>
1159 </sequence>
1160 </complexType>
1161 <element name="AuthnContextClassRef" type="anyURI"/>
1162 <element name="AuthnContextDeclRef" type="anyURI"/>
1163 <element name="AuthnContextDecl" type="anyType"/>
1164 <element name="AuthenticatingAuthority" type="anyURI"/>

```

### 2.7.3 Element <AttributeStatement>

The <AttributeStatement> element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the following elements:

<Attribute> or <EncryptedAttribute> [One or More]

The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML attribute may be included with the <EncryptedAttribute> element.

The following schema fragment defines the <AttributeStatement> element and its **AttributeStatementType** complex type:

```

1176 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1177 <complexType name="AttributeStatementType">
1178     <complexContent>
1179         <extension base="saml:StatementAbstractType">
1180             <choice maxOccurs="unbounded">
1181                 <element ref="saml:Attribute"/>
1182                 <element ref="saml:EncryptedAttribute"/>
1183             </choice>
1184         </extension>
1185     </complexContent>
1186 </complexType>

```

#### 2.7.3.1 Element <Attribute>

The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the **AttributeType** complex type. It is used within an attribute statement to express particular attributes and values associated with an assertion subject, as described in the previous section. It is also used in an attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for more information). The <Attribute> element contains the following XML attributes:

Name [Required]

The name of the attribute.

NameFormat [Optional]

A URI reference representing the classification of the attribute name for purposes of interpreting the

name. See Section 8.2 for some URI references that MAY be used as the value of the `NameFormat` attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section 8.2.1) is in effect.

#### FriendlyName [Optional]

A string that provides a more human-readable form of the attribute's name, which may be useful in cases in which the actual `Name` is complex or opaque, such as an OID or a UUID. This attribute's value MUST NOT be used as a basis for formally identifying SAML attributes.

#### Arbitrary attributes

This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply additional parameters to be used, for example, in an attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

#### <AttributeValue> [Any Number]

Contains a value of the attribute. If an attribute contains more than one discrete value, it is RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have the identical datatype assigned.

The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of values indicates that the requester is interested in any or all of the named attribute's values (see also Section 3.3.2.3).

Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the semantics of specifying or omitting `<AttributeValue>` elements.

The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
  <sequence>
    <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Name" type="string" use="required"/>
  <attribute name="NameFormat" type="anyURI" use="optional"/>
  <attribute name="FriendlyName" type="string" use="optional"/>
  <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

#### 2.7.3.1.1 Element <AttributeValue>

The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data elements MAY be defined in an extension schema.



**Note:** Specifying a datatype other than an XML Schema simple type on `<AttributeValue>` using `xsi:type` will require the presence of the extension schema that defines the datatype in order for schema processing to proceed.

If a SAML attribute includes an empty value, such as the empty string, the corresponding `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`). This overrides the requirement in Section 1.3.1 that string values in SAML content contain at least one non-whitespace character.

If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

The following schema fragment defines the `<AttributeValue>` element:

```
<element name="AttributeValue" type="anyType" nillable="true"/>
```

### 2.7.3.2 Element `<EncryptedAttribute>`

The `<EncryptedAttribute>` element represents a SAML attribute in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAttribute>` element contains the following elements:

`<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

`<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name identifier, as defined by Section 8.3.6.

Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through an intermediary.

The following schema fragment defines the `<EncryptedAttribute>` element:

```
<element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

### 2.7.4 Element `<AuthzDecisionStatement>`

**Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that a request for access by the assertion subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence. Assertions containing `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

The resource is identified by means of a URI reference. In order for the assertion to be interpreted correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different

1284 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
 1285 URI references are to be found in IETF RFC 2396 [RFC 2396] Section 6:

1286 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
 1287 dependent. When a scheme uses elements of the common syntax, it will also use the common  
 1288 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
 1289 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
 1290 the port is elided.

1291 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the  
 1292 URI normalized form wherever possible as follows:

- 1293 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1294 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1295 Inconsistent URI reference interpretation can also result from differences between the URI reference  
 1296 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
 1297 employed to specify an access control policy language. The following security conditions SHOULD be  
 1298 satisfied by the system which employs SAML assertions:

- 1299 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
 1300 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
 1301 part of the resource URI reference.
- 1302 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
 1303 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
 1304 gain access to a denied resource by creating such an equivalence.

1305 The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends  
 1306 **StatementAbstractType** with the addition of the following elements and attributes:

1307 **Resource** [Required]

1308 A URI reference identifying the resource to which access authorization is sought. This attribute MAY  
 1309 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the  
 1310 current document", as specified by IETF RFC 2396 [RFC 2396] Section 4.2.

1311 **Decision** [Required]

1312 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
 1313 the **DecisionType** simple type.

1314 **<Action>** [One or more]

1315 The set of actions authorized to be performed on the specified resource.

1316 **<Evidence>** [Optional]

1317 A set of assertions that the SAML authority relied on in making the decision.

1318 The following schema fragment defines the `<AuthzDecisionStatement>` element and its  
 1319 **AuthzDecisionStatementType** complex type:

```

1320 <element name="AuthzDecisionStatement"
1321   type="saml:AuthzDecisionStatementType"/>
1322 <complexType name="AuthzDecisionStatementType">
1323   <complexContent>
1324     <extension base="saml:StatementAbstractType">
1325       <sequence>
1326         <element ref="saml:Action" maxOccurs="unbounded"/>
1327         <element ref="saml:Evidence" minOccurs="0"/>
1328       </sequence>
1329       <attribute name="Resource" type="anyURI" use="required"/>
1330       <attribute name="Decision" type="saml:DecisionType" use="required"/>
  
```

```

1331     </extension>
1332   </complexContent>
1333 </complexType>

```

#### 2.7.4.1 Simple Type DecisionType

The **DecisionType** simple type defines the possible values to be reported as the status of an authorization decision statement.

Permit

The specified action is permitted.

Deny

The specified action is denied.

Indeterminate

The SAML authority cannot determine whether the specified action is permitted or denied.

The **Indeterminate** decision value is used in situations where the SAML authority requires the ability to provide an affirmative statement but where it is not able to issue a decision. Additional information as to the reason for the refusal or inability to provide a decision MAY be returned as **<StatusDetail>** elements in the enclosing **<Response>**.

The following schema fragment defines the **DecisionType** simple type:

```

1348 <simpleType name="DecisionType">
1349   <restriction base="string">
1350     <enumeration value="Permit"/>
1351     <enumeration value="Deny"/>
1352     <enumeration value="Indeterminate"/>
1353   </restriction>
1354 </simpleType>

```

#### 2.7.4.2 Element <Action>

The **<Action>** element specifies an action on the specified resource for which permission is sought. Its string-data content provides the label for an action sought to be performed on the specified resource, and it has the following attribute:

Namespace [Optional]

A URI reference representing the namespace in which the name of the specified action is to be interpreted. If this element is absent, the namespace

urn:oasis:names:tc:SAML:1.0:action:rwedc-negation specified in Section 8.1.2 is in effect.

The following schema fragment defines the **<Action>** element and its **ActionType** complex type:

```

1365 <element name="Action" type="saml:ActionType"/>
1366 <complexType name="ActionType">
1367   <simpleContent>
1368     <extension base="string">
1369       <attribute name="Namespace" type="anyURI" use="required"/>
1370     </extension>
1371   </simpleContent>
1372 </complexType>

```

### 2.7.4.3 Element <Evidence>

The <Evidence> element contains one or more assertions or assertion references that the SAML authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one or more of the following elements:

<AssertionIDRef> [Any number]

Specifies an assertion by reference to the value of the assertion's ID attribute.

<AssertionURIRef> [Any number]

Specifies an assertion by means of a URI reference.

<Assertion> [Any number]

Specifies an assertion by value.

<EncryptedAssertion> [Any number]

Specifies an encrypted assertion by value.

Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party and the SAML authority making the authorization decision. For example, in the case that the SAML relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use that assertion as evidence in making its authorization decision without endorsing the <Evidence> element's assertion as valid either to the relying party or any other third party.

The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
<element name="Evidence" type="saml:EvidenceType"/>
<complexType name="EvidenceType">
  <choice maxOccurs="unbounded">
    <element ref="saml:AssertionIDRef"/>
    <element ref="saml:AssertionURIRef"/>
    <element ref="saml:Assertion"/>
    <element ref="saml:EncryptedAssertion"/>
  </choice>
</complexType>
```

### 3 SAML Protocols

SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML bindings specification [SAMLBind] describes specific means of transporting protocol messages using existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a number of applications of the protocols defined in this section together with additional processing rules, restrictions, and requirements that facilitate interoperability.

Specific SAML request and response messages derive from common types. The requester sends an element derived from **RequestAbstractType** to a SAML responder, and the responder generates an element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

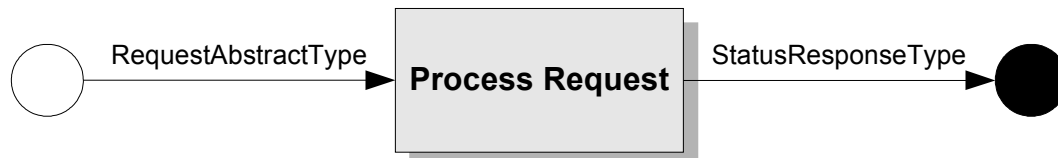


Figure 1: SAML Request-Response Protocol

In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the responder having received a corresponding request.

The protocols defined by SAML achieve the following actions:

- Returning one or more requested assertions. This can occur in response to either a direct request for specific assertions or a query for assertions that meet particular criteria.
- Performing authentication on request and returning the corresponding assertion
- Registering a name identifier or terminating a name registration on request
- Retrieving a protocol message that has been requested by means of an artifact
- Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on request
- Providing a name identifier mapping on request

Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not shown with the conventional namespace prefix `samlp:.` For clarity, text descriptions of elements and types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:.`

#### 3.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the protocol schema:

```

<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  
```

```

1439 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1440   schemaLocation="saml-schema-assertion-2.0.xsd"/>
1441 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1442   schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
1443 20020212/xmldsig-core-schema.xsd"/>
1444 <annotation>
1445   <documentation>
1446     Document identifier: saml-schema-protocol-2.0
1447     Location: http://docs.oasis-open.org/security/saml/v2.0/
1448     Revision history:
1449     V1.0 (November, 2002):
1450       Initial Standard Schema.
1451     V1.1 (September, 2003):
1452       Updates within the same V1.0 namespace.
1453     V2.0 (March, 2005):
1454       New protocol schema based in a SAML V2.0 namespace.
1455   </documentation>
1456 </annotation>
1457 ...
1458 </schema>

```

## 1459 3.2 Requests and Responses

1460 The following sections define the SAML constructs and basic requirements that underlie all of the request  
 1461 and response messages used in SAML protocols.

### 1462 3.2.1 Complex Type RequestAbstractType

1463 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
 1464 This type defines common attributes and elements that are associated with all SAML requests:

1465 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**  
 1466 for V2.0 of SAML.

#### 1467 ID [Required]

1468 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section  
 1469 1.3.4 for identifier uniqueness. The values of the ID attribute in a request and the InResponseTo  
 1470 attribute in the corresponding response MUST match.

#### 1471 Version [Required]

1472 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".  
 1473 SAML versioning is discussed in Section 4.

#### 1474 IssueInstant [Required]

1475 The time instant of issue of the request. The time value is encoded in UTC, as described in Section  
 1476 1.3.3.

#### 1477 Destination [Optional]

1478 A URI reference indicating the address to which this request has been sent. This is useful to prevent  
 1479 malicious forwarding of requests to unintended recipients, a protection that is required by some  
 1480 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the  
 1481 location at which the message was received. If it does not, the request MUST be discarded. Some  
 1482 protocol bindings may require the use of this attribute (see [SAMLBind]).

#### 1483 Consent [Optional]

1484 Indicates whether or not (and under what conditions) consent has been obtained from a principal in  
 1485 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value



1486 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the  
 1487 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in  
 1488 effect.

1489 `<saml:Issuer>` [Optional]

1490 Identifies the entity that generated the request message. (For more information on this element, see  
 1491 Section 2.2.5.)

1492 `<ds:Signature>` [Optional]

1493 An XML Signature that authenticates the requester and provides message integrity, as described  
 1494 below and in Section 5.

1495 `<Extensions>` [Optional]

1496 This extension point contains optional protocol message extension elements that are agreed on  
 1497 between the communicating parties. No extension schema is required in order to make use of this  
 1498 extension point, and even if one is provided, the lax validation setting does not impose a requirement  
 1499 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-  
 1500 SAML-defined namespace.

1501 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to  
 1502 authenticate itself, and message integrity may often be required. Authentication and message integrity  
 1503 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request  
 1504 MAY be signed, which provides both authentication of the requester and message integrity.

1505 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML  
 1506 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)  
 1507 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the  
 1508 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the  
 1509 signature to determine the identity and appropriateness of the signer and may continue to process the  
 1510 request or respond with an error (if the request is invalid for some other reason).

1511 If a `Consent` attribute is included and the value indicates that some form of principal consent has been  
 1512 obtained, then the request SHOULD be signed.

1513 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if  
 1514 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value  
 1515 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a  
 1516 suspected denial-of-service attack, not responding at all may be warranted.

1517 The following schema fragment defines the **RequestAbstractType** complex type:

```

1518 <complexType name="RequestAbstractType" abstract="true">
1519   <sequence>
1520     <element ref="saml:Issuer" minOccurs="0"/>
1521     <element ref="ds:Signature" minOccurs="0"/>
1522     <element ref="samlp:Extensions" minOccurs="0"/>
1523   </sequence>
1524   <attribute name="ID" type="ID" use="required"/>
1525   <attribute name="Version" type="string" use="required"/>
1526   <attribute name="IssueInstant" type="dateTime" use="required"/>
1527   <attribute name="Destination" type="anyURI" use="optional"/>
1528   <attribute name="Consent" type="anyURI" use="optional"/>
1529 </complexType>
1530 <element name="Extensions" type="samlp:ExtensionsType"/>
1531 <complexType name="ExtensionsType">
1532   <sequence>
1533     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1534   </sequence>
1535 </complexType>

```

### 3.2.2 Complex Type **StatusResponseType**

All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type defines common attributes and elements that are associated with all SAML responses:

#### **ID** [Required]

An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness.

#### **InResponseTo** [Optional]

A reference to the identifier of the request to which the response corresponds, if any. If the response is not generated in response to a request, or if the **ID** attribute value of a request cannot be determined (for example, the request is malformed), then this attribute MUST NOT be present. Otherwise, it MUST be present and its value MUST match the value of the corresponding request's **ID** attribute.

#### **Version** [Required]

The version of this response. The identifier for the version of SAML defined in this specification is "2.0". SAML versioning is discussed in Section 4.

#### **IssueInstant** [Required]

The time instant of issue of the response. The time value is encoded in UTC, as described in Section 1.3.3.

#### **Destination** [Optional]

A URI reference indicating the address to which this response has been sent. This is useful to prevent malicious forwarding of responses to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the location at which the message was received. If it does not, the response MUST be discarded. Some protocol bindings may require the use of this attribute (see [SAMLBind]).

#### **Consent** [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this response. See Section 8.4 for some URI references that MAY be used as the value of the **Consent** attribute and their associated descriptions. If no **Consent** value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in effect.

#### **<saml:Issuer>** [Optional]

Identifies the entity that generated the response message. (For more information on this element, see Section 2.2.5.)

#### **<ds:Signature>** [Optional]

An XML Signature that authenticates the responder and provides message integrity, as described below and in Section 5.

#### **<Extensions>** [Optional]

This extension point contains optional protocol message extension elements that are agreed on between the communicating parties. . No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-SAML-defined namespace.

#### **<Status>** [Required]

A code representing the status of the corresponding request.

Depending on the requirements of particular protocols or profiles, a SAML responder may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML response MAY be signed, which provides both authentication of the responder and message integrity.

If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML requester receiving the response MUST verify that the signature is valid (that is, that the message has not been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the response as it deems appropriate.

If a `Consent` attribute is included and the value indicates that some form of principal consent has been obtained, then the response SHOULD be signed.

The following schema fragment defines the **StatusResponseType** complex type:

```
<complexType name="StatusResponseType">
  <sequence>
    <element ref="saml:Issuer" minOccurs="0"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="samlp:Extensions" minOccurs="0"/>
    <element ref="samlp:Status"/>
  </sequence>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="InResponseTo" type="NCName" use="optional"/>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
  <attribute name="Destination" type="anyURI" use="optional"/>
  <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

### 3.2.2.1 Element `<Status>`

The `<Status>` element contains the following elements:

`<StatusCode>` [Required]

A code representing the status of the activity carried out in response to the corresponding request.

`<StatusMessage>` [Optional]

A message which MAY be returned to an operator.

`<StatusDetail>` [Optional]

Additional information concerning the status of the request.

The following schema fragment defines the `<Status>` element and its **StatusType** complex type:

```
<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
  <sequence>
    <element ref="samlp:StatusCode"/>
    <element ref="samlp:StatusMessage" minOccurs="0"/>
    <element ref="samlp:StatusDetail" minOccurs="0"/>
  </sequence>
</complexType>
```

### 3.2.2.2 Element `<StatusCode>`

The `<StatusCode>` element specifies a code or a set of nested codes representing the status of the corresponding request. The `<StatusCode>` element has the following element and attribute:

1627 Value [Required]

1628 The status code value. This attribute contains a URI reference. The value of the topmost  
1629 <StatusCode> element MUST be from the top-level list provided in this section.

1630 <StatusCode> [Optional]

1631 A subordinate status code that provides more specific information on an error condition. Note that  
1632 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for  
1633 additional information by intentionally presenting erroneous requests.

1634 The permissible top-level <StatusCode> values are as follows:

1635 urn:oasis:names:tc:SAML:2.0:status:Success

1636 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or  
1637 <StatusDetail> elements.

1638 urn:oasis:names:tc:SAML:2.0:status:Requester

1639 The request could not be performed due to an error on the part of the requester.

1640 urn:oasis:names:tc:SAML:2.0:status:Responder

1641 The request could not be performed due to an error on the part of the SAML responder or SAML  
1642 authority.

1643 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1644 The SAML responder could not process the request because the version of the request message was  
1645 incorrect.

1646 The following second-level status codes are referenced at various places in this specification. Additional  
1647 second-level status codes MAY be defined in future versions of the SAML specification. System entities  
1648 are free to define more specific status codes by defining appropriate URI references.

1649 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1650 The responding provider was unable to successfully authenticate the principal.

1651 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1652 Unexpected or invalid content was encountered within a <saml:Attribute> or  
1653 <saml:AttributeValue> element.

1654 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1655 The responding provider cannot or will not support the requested name identifier policy.

1656 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1657 The specified authentication context requirements cannot be met by the responder.

1658 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1659 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an  
1660 <IDPList> can be resolved or that none of the supported identity providers are available.

1661 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1662 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1663 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

1664 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are  
1665 supported by the intermediary.

1666 urn:oasis:names:tc:SAML:2.0:status:PartialLogout  
 1667     Used by a session authority to indicate to a session participant that it was not able to propagate logout  
 1668     to all other session participants.

1669 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded  
 1670     Indicates that a responding provider cannot authenticate the principal directly and is not permitted to  
 1671     proxy the request further.

1672 urn:oasis:names:tc:SAML:2.0:status:RequestDenied  
 1673     The SAML responder or SAML authority is able to process the request but has chosen not to respond.  
 1674     This status code MAY be used when there is concern about the security context of the request  
 1675     message or the sequence of request messages received from a particular requester.

1676 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported  
 1677     The SAML responder or SAML authority does not support the request.

1678 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated  
 1679     The SAML responder cannot process any requests with the protocol version specified in the request.

1680 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh  
 1681     The SAML responder cannot process the request because the protocol version specified in the  
 1682     request message is a major upgrade from the highest protocol version supported by the responder.

1683 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow  
 1684     The SAML responder cannot process the request because the protocol version specified in the  
 1685     request message is too low.

1686 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized  
 1687     The resource value provided in the request message is invalid or unrecognized.

1688 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses  
 1689     The response message would contain more elements than the SAML responder is able to return.

1690 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile  
 1691     An entity that has no knowledge of a particular attribute profile has been presented with an attribute  
 1692     drawn from that profile.

1693 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal  
 1694     The responding provider does not recognize the principal specified or implied by the request.

1695 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding  
 1696     The SAML responder cannot properly fulfill the request using the protocol binding specified in the  
 1697     request.

1698 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
 1699 type:

```

1700 <element name="StatusCode" type="samlp:StatusCodeType"/>
1701 <complexType name="StatusCodeType">
1702   <sequence>
1703     <element ref="samlp:StatusCode" minOccurs="0"/>
1704   </sequence>
1705   <attribute name="Value" type="anyURI" use="required"/>
1706 </complexType>

```

### 3.2.2.3 Element <StatusMessage>

The <StatusMessage> element specifies a message that MAY be returned to an operator:

The following schema fragment defines the <StatusMessage> element:

```
<element name="StatusMessage" type="string"/>
```

### 3.2.2.4 Element <StatusDetail>

The <StatusDetail> element MAY be used to specify additional information concerning the status of the request. The additional information consists of zero or more elements from any namespace, with no requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

The following schema fragment defines the <StatusDetail> element and its **StatusDetailType** complex type:

```
<element name="StatusDetail" type="samlp:StatusDetailType"/>
<complexType name="StatusDetailType">
  <sequence>
    <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

## 3.3 Assertion Query and Request Protocol

This section defines messages and processing rules for requesting existing assertions by reference or querying for assertions by subject and statement type.

### 3.3.1 Element <AssertionIDRequest>

If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest> message element can be used to request that they be returned in a <Response> message. The <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for more information on this element.

The following schema fragment defines the <AssertionIDRequest> element:

```
<element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
<complexType name="AssertionIDRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.2 Queries

The following sections define the SAML query request messages.

#### 3.3.2.1 Element <SubjectQuery>

The <SubjectQuery> message element is an extension point that allows new SAML queries to be defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and



is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the `<saml:Subject>` element (defined in Section 2.4) to **RequestAbstractType**.

The following schema fragment defines the `<SubjectQuery>` element and its **SubjectQueryAbstractType** complex type:

```
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="saml:Subject"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.2.2 Element `<AuthnQuery>`

The `<AuthnQuery>` message element is used to make the query "What assertions containing authentication statements are available for this subject?" A successful `<Response>` will contain one or more assertions containing authentication statements.

The `<AuthnQuery>` message MUST NOT be used as a request for a new authentication using credentials provided in the request. `<AuthnQuery>` is a request for statements about authentication acts that have occurred in a previous interaction between the indicated subject and the authentication authority.

This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

**SessionIndex** [Optional]

If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject within the context of the supplied session information?"

**<RequestedAuthnContext>** [Optional]

If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject that satisfy the authentication context requirements in this element?"

In response to an authentication query, a SAML authority returns assertions with authentication statements as follows:

- Rules given in Section 3.3.4 for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If the **SessionIndex** attribute is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions MUST contain a **SessionIndex** attribute that matches the **SessionIndex** attribute in the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.
- If the `<RequestedAuthnContext>` element is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions MUST contain an `<AuthnContext>` element that satisfies the element in the query (see Section 3.3.2.2.1). It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.

The following schema fragment defines the `<AuthnQuery>` element and its **AuthnQueryType** complex type:

```
<element name="AuthnQuery" type="samlp:AuthnQueryType"/>
```

```

1794 <complexType name="AuthnQueryType">
1795   <complexContent>
1796     <extension base="samlp:SubjectQueryAbstractType">
1797       <sequence>
1798         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1799       </sequence>
1800       <attribute name="SessionIndex" type="string" use="optional"/>
1801     </extension>
1802   </complexContent>
1803 </complexType>

```

### 1804 3.3.2.2.1 Element <RequestedAuthnContext>

1805 The <RequestedAuthnContext> element specifies the authentication context requirements of  
 1806 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**  
 1807 complex type defines the following elements and attributes:

1808 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1809 Specifies one or more URI references identifying authentication context classes or declarations.  
 1810 These elements are defined in Section 2.7.2.2. For more information about authentication context  
 1811 classes, see [SAMLAuthnCxt].

1812 Comparison [Optional]

1813 Specifies the comparison method used to evaluate the requested context classes or statements, one  
 1814 of "exact", "minimum", "maximum", or "better". The default is "exact".

1815 Either a set of class references or a set of declaration references can be used. The set of supplied  
 1816 references MUST be evaluated as an ordered set, where the first element is the most preferred  
 1817 authentication context class or declaration. If none of the specified classes or declarations can be satisfied  
 1818 in accordance with the rules below, then the responder MUST return a <Response> message with a  
 1819 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1820 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
 1821 statement MUST be the exact match of at least one of the authentication contexts specified.

1822 If Comparison is set to "minimum", then the resulting authentication context in the authentication  
 1823 statement MUST be at least as strong (as deemed by the responder) as one of the authentication  
 1824 contexts specified.

1825 If Comparison is set to "better", then the resulting authentication context in the authentication  
 1826 statement MUST be stronger (as deemed by the responder) than any one of the authentication contexts  
 1827 specified.

1828 If Comparison is set to "maximum", then the resulting authentication context in the authentication  
 1829 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength  
 1830 of at least one of the authentication contexts specified.

1831 The following schema fragment defines the <RequestedAuthnContext> element and its  
 1832 **RequestedAuthnContextType** complex type:

```

1833 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1834 <complexType name="RequestedAuthnContextType">
1835   <choice>
1836     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1837     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1838   </choice>
1839   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1840 use="optional"/>
1841 </complexType>
1842 <simpleType name="AuthnContextComparisonType">

```

```

1843     <restriction base="string">
1844         <enumeration value="exact"/>
1845         <enumeration value="minimum"/>
1846         <enumeration value="maximum"/>
1847         <enumeration value="better"/>
1848     </restriction>
1849 </simpleType>

```

### 3.3.2.3 Element <AttributeQuery>

The <AttributeQuery> element is used to make the query "Return the requested attributes for this subject." A successful response will be in the form of assertions containing attribute statements, to the extent allowed by policy. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element:

<saml:Attribute> [Any Number]

Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no attributes are specified, it indicates that all attributes allowed by policy are requested. If a given <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if that attribute is returned in the response, it MUST NOT contain any values that are not equal to the values specified in the query. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on <saml:Attribute>, see Section 2.7.3.1.

A single query MUST NOT contain two <saml:Attribute> elements with the same Name and NameFormat values (that is, a given attribute MUST be named only once in a query).

In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the assertions that may be returned.
- If any <Attribute> elements are present in the query, they constrain/filter the attributes and optionally the values returned, as noted above.
- The attributes and values returned MAY also be constrained by application-specific policy considerations.

The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to indicate problems with the interpretation of attribute or value information in a query.

The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType** complex type:

```

1877 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1878 <complexType name="AttributeQueryType">
1879     <complexContent>
1880         <extension base="samlp:SubjectQueryAbstractType">
1881             <sequence>
1882                 <element ref="saml:Attribute" minOccurs="0"
1883 maxOccurs="unbounded"/>
1884             </sequence>
1885         </extension>
1886     </complexContent>
1887 </complexType>

```

### 3.3.2.4 Element <AuthzDecisionQuery>

The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource be allowed for this subject, given this evidence?” A successful response will be in the form of assertions containing authorization decision statements.

**Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following elements and attribute:

**Resource** [Required]

A URI reference indicating the resource for which authorization is requested.

**<saml:Action>** [One or More]

The actions for which authorization is requested. For more information on this element, see Section 2.7.4.2.

**<saml:Evidence>** [Optional]

A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more information on this element, see Section 2.7.4.3.

In response to an authorization decision query, a SAML authority returns assertions with authorization decision statements as follows:

- Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the assertions that may be returned.

The following schema fragment defines the <AuthzDecisionQuery> element and its **AuthzDecisionQueryType** complex type:

```
<element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
<complexType name="AuthzDecisionQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
        <element ref="saml:Evidence" minOccurs="0"/>
      </sequence>
      <attribute name="Resource" type="anyURI" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.3 Element <Response>

The <Response> message element is used when a response consists of a list of zero or more assertions that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType** and adds the following elements:

**<saml:Assertion>** or **<saml:EncryptedAssertion>** [Any Number]

Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for more information on these elements.

The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1932 <element name="Response" type="samlp:ResponseType"/>
1933 <complexType name="ResponseType">
1934   <complexContent>
1935     <extension base="samlp:StatusResponseType">
1936       <choice minOccurs="0" maxOccurs="unbounded">
1937         <element ref="saml:Assertion"/>
1938         <element ref="saml:EncryptedAssertion"/>
1939       </choice>
1940     </extension>
1941   </complexContent>
1942 </complexType>

```

### 3.3.4 Processing Rules

In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST** contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the query.

A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both apply:

- If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST** include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2. In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical" means that the identifier element's content and attribute values **MUST** be the same. An encrypted identifier will be identical to the original according to this definition, once decrypted.
- If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner described by at least one `<saml:SubjectConfirmation>` element in S2.

As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with different `Format` values and element content, even if the two identifiers are considered to refer to the same principal.

If the SAML authority cannot provide an assertion with any statements satisfying the constraints expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Success`.

All other processing rules associated with the underlying request and response messages **MUST** be observed.

## 3.4 Authentication Request Protocol

When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing authentication statements to establish a security context at one or more relying parties, it can use the authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and request that it return a `<Response>` message containing one or more such assertions. Such assertions **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

Apart from this requirement, the specific contents of the returned assertions depend on the profile or context of use. Also, the exact means by which the principal or agent authenticates to the identity provider is not specified, though the means of authentication might impact the content of the response. Other issues related to the validation of authentication credentials by the identity provider or any communication

1979 between the identity provider and any other entities involved in the authentication process are also out of  
1980 scope of this protocol.

1981 The descriptions and processing rules in the following sections reference the following actors, many of  
1982 whom might be the same entity in a particular profile of use:

1983 Requester

1984 The entity who creates the authentication request and to whom the response is to be returned.

1985 Presenter

1986 The entity who presents the request to the identity provider and either authenticates itself during  
1987 the transmission of the message, or relies on an existing security context to establish its identity. If  
1988 not the requester, the presenter acts as an intermediary between the requester and the  
1989 responding identity provider.

1990 Requested Subject

1991 The entity about whom one or more assertions are being requested.

1992 Attesting Entity

1993 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`  
1994 elements of the resulting assertion(s).

1995 Relying Party

1996 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1997 the profile or context of use, generally to establish a security context.

1998 Identity Provider

1999 The entity to whom the presenter gives the request and from whom the presenter receives the  
2000 response.

2001 **3.4.1 Element `<AuthnRequest>`**

2002 To request that an identity provider issue an assertion with an authentication statement, a presenter  
2003 authenticates to that identity provider (or relies on an existing security context) and sends it an  
2004 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to  
2005 satisfy its purpose. Among these properties may be information that relates to the content of the assertion  
2006 and/or information that relates to how the resulting `<Response>` message should be delivered to the  
2007 requester. The process of authentication of the presenter may take place before, during, or after the initial  
2008 delivery of the `<AuthnRequest>` message.

2009 The requester might not be the same as the presenter of the request if, for example, the requester is a  
2010 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject  
2011 so that the relying party can decide whether to provide a service.

2012 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected  
2013 by the protocol binding used to deliver the message.

2014 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
2015 adds the following elements and attributes, all of which are optional in general, but may be required by  
2016 specific profiles:

2017 `<saml:Subject>` [Optional]

2018 Specifies the requested subject of the resulting assertion(s). This may include one or more  
2019 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions  
2020 can be confirmed. For more information on this element, see Section 2.4.



2021 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
 2022 requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the presenter  
 2023 is presumed to be the only attesting entity required and the method is implied by the profile of use  
 2024 and/or the policies of the identity provider.

2025 `<NameIDPolicy>` [Optional]

2026 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,  
 2027 then any type of identifier supported by the identity provider for the requested subject can be used,  
 2028 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2029 `<saml:Conditions>` [Optional]

2030 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting  
 2031 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The  
 2032 information in this element is used as input to the process of constructing the assertion, rather than as  
 2033 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2034 `<RequestedAuthnContext>` [Optional]

2035 Specifies the requirements, if any, that the requester places on the authentication context that applies  
 2036 to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing rules  
 2037 regarding this element.

2038 `<Scoping>` [Optional]

2039 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as  
 2040 limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity  
 2041 providers by the responder.

2042 `ForceAuthn` [Optional]

2043 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
 2044 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
 2045 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the  
 2046 presenter unless the constraints of `IsPassive` can be met.

2047 `IsPassive` [Optional]

2048 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take control  
 2049 of the user interface from the requester and interact with the presenter in a noticeable fashion. If a  
 2050 value is not provided, the default is "false".

2051 `AssertionConsumerServiceIndex` [Optional]

2052 Indirectly identifies the location to which the `<Response>` message should be returned to the  
 2053 requester. It applies only to profiles in which the requester is different from the presenter, such as the  
 2054 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map  
 2055 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one  
 2056 possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message to  
 2057 the default location associated with the requester for the profile of use. If the index specified is invalid,  
 2058 then the identity provider MAY return an error `<Response>` or it MAY use the default location. This  
 2059 attribute is mutually exclusive with the `AssertionConsumerServiceURL` and `ProtocolBinding`  
 2060 attributes.

2061 `AssertionConsumerServiceURL` [Optional]

2062 Specifies by value the location to which the `<Response>` message MUST be returned to the  
 2063 requester. The responder MUST ensure by some means that the value specified is in fact associated  
 2064 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing  
 2065 `<AuthnRequest>` message is another. This attribute is mutually exclusive with the  
 2066 `AssertionConsumerServiceIndex` attribute and is typically accompanied by the  
 2067 `ProtocolBinding` attribute.

## 2068 ProtocolBinding [Optional]

2069 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>  
 2070 message. See [SAMLBind] for more information about protocol bindings and URI references defined  
 2071 for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute  
 2072 and is typically accompanied by the AssertionConsumerServiceURL attribute.

## 2073 AttributeConsumingServiceIndex [Optional]

2074 Indirectly identifies information associated with the requester describing the SAML attributes the  
 2075 requester desires or requires to be supplied by the identity provider in the <Response> message. The  
 2076 identity provider MUST have a trusted means to map the index value in the attribute to information  
 2077 associated with the requester. [SAMLMeta] provides one possible mechanism. The identity provider  
 2078 MAY use this information to populate one or more <saml:AttributeStatement> elements in the  
 2079 assertion(s) it returns.

## 2080 ProviderName [Optional]

2081 Specifies the human-readable name of the requester for use by the presenter's user agent or the  
 2082 identity provider.

2083 See Section 3.4.1.4 for general processing rules regarding this message.

2084 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**  
 2085 complex type:

```

2086 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2087 <complexType name="AuthnRequestType">
2088   <complexContent>
2089     <extension base="samlp:RequestAbstractType">
2090       <sequence>
2091         <element ref="saml:Subject" minOccurs="0"/>
2092         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2093         <element ref="saml:Conditions" minOccurs="0"/>
2094         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2095         <element ref="samlp:Scoping" minOccurs="0"/>
2096       </sequence>
2097       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2098       <attribute name="IsPassive" type="boolean" use="optional"/>
2099       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2100       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2101 use="optional"/>
2102       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2103 use="optional"/>
2104       <attribute name="AttributeConsumingServiceIndex"
2105 type="unsignedShort" use="optional"/>
2106       <attribute name="ProviderName" type="string" use="optional"/>
2107     </extension>
2108   </complexContent>
2109 </complexType>

```

## 2110 3.4.1.1 Element &lt;NameIDPolicy&gt;

2111 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
 2112 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

## 2113 Format [Optional]

2114 Specifies the URI reference corresponding to a name identifier format defined in this or another  
 2115 specification (see Section 8.3 for examples). The additional value of  
 2116 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use  
 2117 within this attribute to indicate a request that the resulting identifier be encrypted.

## 2118 SPNameQualifier [Optional]

2119 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of  
 2120 a service provider other than the requester, or in the namespace of an affiliation group of service  
 2121 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2122 `format:persistent` in Section 8.3.7.

## 2123 AllowCreate [Optional]

2124 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the  
 2125 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the  
 2126 requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for  
 2127 the principal has already been established. Note that this does not prevent the identity provider from  
 2128 creating such identifiers outside the context of this specific request (for example, in advance for a  
 2129 large number of principals).

2130 When this element is used, if the content is not understood by or acceptable to the identity provider, then a  
 2131 `<Response>` message element MUST be returned with an error `<Status>`, and MAY contain a second-  
 2132 level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2133 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2134 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any  
 2135 additional constraints due to the content of this element or the policies of the identity provider or principal.

2136 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates  
 2137 that the resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext. The  
 2138 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the  
 2139 requested subject.

2140 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an  
 2141 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly  
 2142 specific to the service provider) require that an encrypted identifier be used.

2143 Note that if the requester wishes to permit the identity provider to establish a new identifier for the principal  
 2144 if none exists, it MUST include this element with the `AllowCreate` attribute set to "true". Otherwise,  
 2145 only a principal for whom the identity provider has previously established an identifier usable by the  
 2146 requester can be authenticated successfully. This is primarily useful in conjunction with the  
 2147 `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value (see Section 8.3.7).

2148 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**  
 2149 complex type:

```
2150 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
2151 <complexType name="NameIDPolicyType">
2152   <attribute name="Format" type="anyURI" use="optional"/>
2153   <attribute name="SPNameQualifier" type="string" use="optional"/>
2154   <attribute name="AllowCreate" type="boolean" use="optional"/>
2155 </complexType>
```

2156 **3.4.1.2 Element <Scoping>**

2157 The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the  
 2158 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to  
 2159 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
 2160 elements and attribute:

## 2161 ProxyCount [Optional]

2162 Specifies the number of proxying indirections permissible between the identity provider that receives  
 2163 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of  
 2164 zero permits no proxying, while omitting this attribute expresses no such restriction.

2165 <IDPList> [Optional]

2166 An advisory list of identity providers and associated information that the requester deems acceptable  
2167 to respond to the request.

2168 <RequesterID> [Zero or More]

2169 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate  
2170 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for a  
2171 description of entity identifiers.

2172 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
2173 <Response> message with an error <Status> and a second-level <StatusCode> of  
2174 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or  
2175 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support  
2176 any of the specified identity providers.

2177 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
2178 <element name="Scoping" type="samlp:ScopingType"/>
2179 <complexType name="ScopingType">
2180   <sequence>
2181     <element ref="samlp:IDPList" minOccurs="0"/>
2182     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2183   </sequence>
2184   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2185 </complexType>
2186 <element name="RequesterID" type="anyURI"/>
```

### 2187 3.4.1.3 Element <IDPList>

2188 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the  
2189 presenter. Its **IDPListType** complex type defines the following elements:

2190 <IDPEntry> [One or More]

2191 Information about a single identity provider.

2192 <GetComplete> [Optional]

2193 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to  
2194 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML  
2195 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>  
2196 element.

2197 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
2198 <element name="IDPList" type="samlp:IDPListType"/>
2199 <complexType name="IDPListType">
2200   <sequence>
2201     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2202     <element ref="samlp:GetComplete" minOccurs="0"/>
2203   </sequence>
2204 </complexType>
2205 <element name="GetComplete" type="anyURI"/>
```

#### 2206 3.4.1.3.1 Element <IDPEntry>

2207 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the  
2208 presenter. Its **IDPEntryType** complex type defines the following attributes:

2209 ProviderID [Required]

2210 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2211 Name [Optional]

2212 A human-readable name for the identity provider.

2213 Loc [Optional]

2214 A URI reference representing the location of a profile-specific endpoint supporting the authentication  
2215 request protocol. The binding to be used must be understood from the profile of use.

2216 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
2217 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2218 <complexType name="IDPEntryType">
2219   <attribute name="ProviderID" type="anyURI" use="required"/>
2220   <attribute name="Name" type="string" use="optional"/>
2221   <attribute name="Loc" type="anyURI" use="optional"/>
2222 </complexType>
```

### 2223 3.4.1.4 Processing Rules

2224 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
2225 therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
2226 kinds of input and output are required or prohibited. The following processing rules apply as invariant  
2227 behavior across any profile of this protocol exchange. All other processing rules associated with the  
2228 underlying request and response messages **MUST** also be observed.

2229 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing  
2230 one or more assertions that meet the specifications defined by the request, or with a <Response>  
2231 message containing a <Status> describing the error that occurred. The responder **MAY** conduct  
2232 additional message exchanges with the presenter as needed to initiate or complete the authentication  
2233 process, subject to the nature of the protocol binding and the authentication mechanism. As described in  
2234 the next section, this includes proxying the request by directing the presenter to another identity provider  
2235 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to  
2236 authenticate the presenter to the original responder, in effect using SAML as the authentication  
2237 mechanism.

2238 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if  
2239 prevented from providing an assertion by policies in effect at the identity provider (for example the  
2240 intended subject has prohibited the identity provider from providing assertions to the relying party), then it  
2241 **MUST** return a <Response> with an error <Status>, and **MAY** return a second-level <StatusCode> of  
2242 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or  
2243 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2244 If the <saml:Subject> element in the request is present, then the resulting assertions'  
2245 <saml:Subject> **MUST strongly match** the request <saml:Subject>, as described in Section 3.3.4,  
2246 except that the identifier **MAY** be in a different format if specified by <NameIDPolicy>. In such a case,  
2247 the identifier's physical content **MAY** be different, but it **MUST** refer to the same principal.

2248 All of the content defined specifically within <AuthnRequest> is optional, although some may be required  
2249 by certain profiles. In the absence of any specific content at all, the following behavior is implied:

- 2250 • The assertion(s) returned **MUST** contain a <saml:Subject> element that represents the  
2251 presenter. The identifier type and format are determined by the identity provider. At least one  
2252 statement in at least one assertion **MUST** be a <saml:AuthnStatement> that describes the  
2253 authentication performed by the responder or authentication service associated with it.
- 2254 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the  
2255 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation  
2256 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.



- The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element referencing the requester as an acceptable relying party. Other audiences MAY be included as deemed appropriate by the identity provider.

### 3.4.1.5 Proxying

If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or cannot directly authenticate the presenter, but believes that the presenter has already authenticated to another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying identity provider.

Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`, completing the overall message exchange. Both the proxying and authenticating identity providers MAY include constraints on proxying activity in the messages and assertions they issue, as described in previous sections and below.

The requester can influence proxy behavior by including a `<Scoping>` element where the provider sets a desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by including an ordered `<IDPList>` of preferred providers.

An identity provider can control secondary use of its assertions by proxying identity providers using a `<ProxyRestriction>` element in the assertions it issues.

#### 3.4.1.5.1 Proxying Processing Rules

An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly authenticate the presenter.

If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying identity provider MUST include equivalent or stricter forms of all the information included in the original request (such as authentication context policy). Note, however, that the proxying provider is free to specify whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for example) will be honored by the authenticating provider.

The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new request SHOULD contain a `<ProxyCount>` attribute.

If an `<IDPList>` was specified in the original request, the new request MUST also contain an `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the `<IDPList>`, but MUST NOT remove any from the list.



2299 The authentication request and response are processed in normal fashion, in accordance with the rules  
 2300 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity  
 2301 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2302 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
 2303 relevant information from the original assertion or non-SAML equivalent.
- 2304 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original  
 2305 requester's preferences, as defined by its <NameIDPolicy> element.
- 2306 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>  
 2307 element containing a <saml:AuthenticatingAuthority> element referencing the identity  
 2308 provider to which the proxying identity provider referred the presenter. If the original assertion  
 2309 contains <saml:AuthnContext> information that includes one or more  
 2310 <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the  
 2311 new assertion, with the new element placed after them.
- 2312 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
 2313 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
 2314 consistent over time across different requests. The value MUST not conflict with values used or  
 2315 generated by other SAML providers.
- 2316 • Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in  
 2317 accordance with the policies of the proxying identity provider, provided that the original  
 2318 requirements dictated by the requester are met.

2319 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,  
 2320 and this request is equally or less strict than the original request (as determined by the proxying identity  
 2321 provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating  
 2322 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML  
 2323 equivalent it received is still valid).

### 2324 **3.5 Artifact Resolution Protocol**

2325 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be  
 2326 transported in a SAML binding by reference instead of by value. Both requests and responses can be  
 2327 obtained by reference using this specialized protocol. A message sender, instead of binding a message to  
 2328 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a  
 2329 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver  
 2330 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
 2331 protocol to resolve the artifact into the original protocol message.

2332 The most common use for this mechanism is with bindings that cannot easily carry a message because of  
 2333 size constraints, or to enable a message to be communicated via a secure channel between the SAML  
 2334 requester and responder, avoiding the need for a signature.

2335 Depending on the characteristics of the underlying message being passed by reference, the artifact  
 2336 resolution protocol MAY require protections such as mutual authentication, integrity protection,  
 2337 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST  
 2338 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used  
 2339 by any party.

2340 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly  
 2341 as if the message so obtained had been sent originally in place of the artifact.

### 3.5.1 Element <ArtifactResolve>

The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message. The original transmission of the artifact is governed by the specific protocol binding that is being used; see [SAMLBind] for more information on the use of artifacts in bindings.

The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and adds the following element:

<Artifact> [Required]

The artifact value that the requester received and now wishes to translate into the protocol message it represents. See [SAMLBind] for specific artifact format information.

The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType** complex type:

```
<element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
<complexType name="ArtifactResolveType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="samlp:Artifact"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="Artifact" type="string"/>
```

### 3.5.2 Element <ArtifactResponse>

The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse> message element. This element is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol message being returned. This wrapped message element can be a request or a response.

The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ArtifactResponse> element and its **ArtifactResponseType** complex type:

```
<element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
<complexType name="ArtifactResponseType">
  <complexContent>
    <extension base="samlp:StatusResponseType">
      <sequence>
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.5.3 Processing Rules

If the responder recognizes the artifact as valid, then it responds with the associated protocol message in an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>

element with no embedded message. In both cases, the `<Status>` element MUST include a `<StatusCode>` element with the code value `urn:oasis:names:tc:SAML:2.0:status:Success`. A response message with no embedded message inside it is termed an empty response in the remainder of this section.

The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent request with the same artifact by any requester results in an empty response as described above.

Some SAML protocol messages, most particularly the `<AuthnRequest>` message in some profiles, MAY be intended for consumption by any party that receives it and can respond appropriately. In most other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST be associated with the intended recipient of the message that the artifact represents. If the artifact issuer receives an `<ArtifactResolve>` message from a requester that cannot authenticate itself as the original intended recipient, then the artifact issuer MUST return an empty response.

The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and return it in an `<ArtifactResolve>` message to the issuer.

Note that the `<ArtifactResponse>` message's `InResponseTo` attribute MUST contain the value of the corresponding `<ArtifactResolve>` message's `ID` attribute, but the embedded protocol message will contain its own message identifier, and in the case of an embedded response, may contain a different `InResponseTo` value that corresponds to the original request message to which the embedded message is responding.

All other processing rules associated with the underlying request and response messages MUST be observed.

## 3.6 Name Identifier Management Protocol

After establishing a name identifier for a principal, an identity provider wishing to change the value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will no longer be used to refer to the principal, informs service providers of the change by sending them a `<ManageNameIDRequest>` message.

A service provider also uses this message to register or change the `SPProvidedID` value to be included when the underlying name identifier is used to communicate with it, or to terminate the use of a name identifier between itself and the identity provider.

Note that this protocol is typically not used with "transient" name identifiers, since their value is not intended to be managed on a long term basis.

### 3.6.1 Element `<ManageNameIDRequest>`

A provider sends a `<ManageNameIDRequest>` message to inform the recipient of a changed name identifier or to indicate the termination of the use of a name identifier.

The `<ManageNameIDRequest>` message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ManageNameIDRequestType**, which extends **RequestAbstractType** and adds the following elements:

`<saml:NameID>` or `<saml:EncryptedID>` [Required]

The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request. (For more information on these elements, see Section 2.2.)

<NewID> or <NewEncryptedID> or <Terminate> [Required]

The new identifier value (in plaintext or encrypted form) to be used when communicating with the requesting provider concerning this principal, or an indication that the use of the old identifier has been terminated. In the former case, if the requester is the service provider, the new identifier **MUST** appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the identity provider, the new value will appear in subsequent <NameID> elements as the element's content.

The following schema fragment defines the <ManageNameIDRequest> element and its **ManageNameIDRequestType** complex type:

```
<element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
<complexType name="ManageNameIDRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <choice>
          <element ref="saml:NameID"/>
          <element ref="saml:EncryptedID"/>
        </choice>
        <choice>
          <element ref="samlp:NewID"/>
          <element ref="samlp:NewEncryptedID"/>
          <element ref="samlp:Terminate"/>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="NewID" type="string"/>
<element name="NewEncryptedID" type="saml:EncryptedElementType"/>
<element name="Terminate" type="samlp:TerminateType"/>
<complexType name="TerminateType"/>
```

### 3.6.2 Element <ManageNameIDResponse>

The recipient of a <ManageNameIDRequest> message **MUST** respond with a <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional content.

The <ManageNameIDResponse> message **SHOULD** be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ManageNameIDResponse> element:

```
<element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

### 3.6.3 Processing Rules

If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize, the responding provider **MUST** respond with an error <Status> and **MAY** respond with a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

If the <Terminate> element is included in the request, the requesting provider is indicating that (in the case of a service provider) it will no longer accept assertions from the identity provider or (in the case of an identity provider) it will no longer issue assertions to the service provider about the principal. The receiving provider can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for whom a relationship has been terminated.

2481 If the service provider requests that its identifier for the principal be changed by including a `<NewID>` (or  
 2482 `<NewEncryptedID>`) element, the identity provider MUST include the element's content as the  
 2483 `SPProvidedID` when subsequently communicating to the service provider regarding this principal.

2484 If the identity provider requests that its identifier for the principal be changed by including a `<NewID>` (or  
 2485 `<NewEncryptedID>`) element, the service provider MUST use the element's content as the  
 2486 `<saml:NameID>` element content when subsequently communicating with the identity provider regarding  
 2487 this principal.

2488 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the  
 2489 `<EncryptedID>` and `<NewEncryptedID>` elements).

2490 In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute  
 2491 MUST contain the most recent name identifier information established between the providers for the  
 2492 principal.

2493 In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2494 `format:persistent`, the `NameQualifier` attribute MUST contain the unique identifier of the identity  
 2495 provider that created the identifier. If the identifier was established between the identity provider and an  
 2496 affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute MUST  
 2497 contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the  
 2498 service provider. These attributes MAY be omitted if they would otherwise match the value of the  
 2499 containing protocol message's `<Issuer>` element, but this is NOT RECOMMENDED due to the  
 2500 opportunity for confusion.

2501 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
 2502 systems at both the requester and the responder. Implementations might wish to allow each party to  
 2503 accept either identifier for some period of time following the successful completion of a name identifier  
 2504 change. Not doing so could result in the inability of the principal to access resources.

2505 All other processing rules associated with the underlying request and response messages MUST be  
 2506 observed.

### 2507 **3.7 Single Logout Protocol**

2508 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
 2509 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
 2510 when a principal logs out at a session participant or when the principal logs out directly at the  
 2511 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for  
 2512 the logout event can be indicated through the `Reason` attribute.

2513  
 2514 The principal may have established authenticated sessions with both the session authority and individual  
 2515 session participants, based on assertions containing authentication statements supplied by the session  
 2516 authority.

2517  
 2518 When the principal invokes the single logout process at a session participant, the session participant  
 2519 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion  
 2520 containing the authentication statement related to that session at the session participant.

2521  
 2522 When either the principal invokes a logout at the session authority, or a session participant sends a logout  
 2523 request to the session authority specifying that principal, the session authority SHOULD send a  
 2524 `<LogoutRequest>` message to each session participant to which it provided assertions containing  
 2525 authentication statements under its current session with the principal, with the exception of the session  
 2526 participant that sent the `<LogoutRequest>` message to the session authority. It SHOULD attempt to  
 2527 contact as many of these participants as it can using this protocol, terminate its own session with the  
 2528 principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

### 3.7.1 Element <LogoutRequest>

A session participant or session authority sends a <LogoutRequest> message to indicate that a session has been terminated.

The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and adds the following elements and attributes:

**NotOnOrAfter** [Optional]

The time at which the request expires, after which the recipient may discard the message. The time value is encoded in UTC, as described in Section 1.3.3.

**Reason** [Optional]

An indication of the reason for the logout, in the form of a URI reference.

<saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request. (For more information on this element, see Section 2.2.)

<SessionIndex> [Optional]

The identifier that indexes this session at the message recipient.

The following schema fragment defines the <LogoutRequest> element and associated **LogoutRequestType** complex type:

```
<element name="LogoutRequest" type="samlp:LogoutRequestType"/>
<complexType name="LogoutRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <choice>
          <element ref="saml:BaseID"/>
          <element ref="saml:NameID"/>
          <element ref="saml:EncryptedID"/>
        </choice>
        <element ref="samlp:SessionIndex" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="Reason" type="string" use="optional"/>
      <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
    </extension>
  </complexContent>
</complexType>
<element name="SessionIndex" type="string"/>
```

### 3.7.2 Element <LogoutResponse>

The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of type **StatusResponseType**, with no additional content specified.

The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <LogoutResponse> element:



2575 `<element name="LogoutResponse" type="samlp:StatusResponseType"/>`

### 2576 3.7.3 Processing Rules

2577 The message sender MAY use the `Reason` attribute to indicate the reason for sending the  
 2578 `<LogoutRequest>`. The following values are defined by this specification for use by all message  
 2579 senders; other values MAY be agreed on between participants:

2580 `urn:oasis:names:tc:SAML:2.0:logout:user`

2581 Specifies that the message is being sent because the principal wishes to terminate the indicated  
 2582 session.

2583 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2584 Specifies that the message is being sent because an administrator wishes to terminate the indicated  
 2585 session for that principal.

2586 All other processing rules associated with the underlying request and response messages MUST be  
 2587 observed.

2588 Additional processing rules are provided in the following sections.

#### 2589 3.7.3.1 Session Participant Rules

2590 When a session participant receives a `<LogoutRequest>` message, the session participant MUST  
 2591 authenticate the message. If the sender is the authority that provided an assertion containing an  
 2592 authentication statement linked to the principal's current session, the session participant MUST invalidate  
 2593 the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or  
 2594 `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no  
 2595 `<SessionIndex>` elements are supplied, then all sessions associated with the principal MUST be  
 2596 invalidated.

2597 The session participant MUST apply the logout request message to any assertion that meets the following  
 2598 conditions, even if the assertion arrives after the logout request:

- 2600 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or  
 2601 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in Section 3.3.4.
- 2602 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of  
 2603 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no  
 2604 `<SessionIndex>` elements.
- 2605 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself  
 2606 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject  
 2607 confirmation data).
- 2608 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on  
 2609 the message).

2610 **Note:** This rule is intended to prevent a situation in which a session participant receives a  
 2611 logout request targeted at a single, or multiple, assertion(s) (as identified by the  
 2612 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid –  
 2613 assertion(s) targeted by the logout request. It should honor the logout request until the  
 2614 logout request itself may be discarded (the `NotOnOrAfter` value on the request has  
 2615 been exceeded) or the assertion targeted by the logout request has been received and  
 2616 has been handled appropriately.

### 3.7.3.2 Session Authority Rules

When a session authority receives a `<LogoutRequest>` message, the session authority **MUST** authenticate the sender. If the sender is a session participant to which the session authority provided an assertion containing an authentication statement for the current session, then the session authority **SHOULD** do the following in the specified order:

- Send a `<LogoutRequest>` message to any session authority on behalf of whom the session authority proxied the principal's authentication, unless the second authority is the originator of the `<LogoutRequest>`.
- Send a `<LogoutRequest>` message to each session participant for which the session authority provided assertions in the current session, *other than* the originator of a current `<LogoutRequest>`.
- Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout request message.

If the session authority successfully terminates the principal's session with respect to itself, then it **MUST** respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it **MUST** respond with a `<LogoutResponse>` message containing a top-level status code indicating the error. Thus, the top-level status indicates the state of the logout operation only with respect to the session authority itself.

The session authority **SHOULD** attempt to contact each session participant using any applicable/usable protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because the original request takes place using a protocol binding that does not enable the logout to be propagated to all participants).

In the event that not all session participants successfully respond to these `<LogoutRequest>` messages (or if not all participants can be contacted), then the session authority **MUST** include in its `<LogoutResponse>` message a second-level status code of `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session participants successfully responded with confirmation of the logout.

Note that a session authority **MAY** initiate a logout for reasons other than having received a `<LogoutRequest>` from a session participant – these include, but are not limited to:

- If some timeout period was agreed out-of-band with an individual session participant, the session authority **MAY** send a `<LogoutRequest>` to that individual participant alone.
- An agreed global timeout period has been exceeded.
- The principal or some other trusted entity has requested logout of the principal directly at the session authority.
- The session authority has determined that the principal's credentials may have been compromised.

When constructing a logout request message, the session authority **MUST** set the value of the `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message, after which the logout request may be discarded by the recipient. This value **SHOULD** be set to a time value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout request).

In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also available for use by the session authority only:

`urn:oasis:names:tc:SAML:2.0:logout:global-timeout`

2663 Specifies that the message is being sent because of the global session timeout interval period  
 2664 being exceeded.

2665 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2666 Specifies that the message is being sent because a timeout interval period agreed between a  
 2667 participant and the session authority has been exceeded.

## 2668 3.8 Name Identifier Mapping Protocol

2669 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
 2670 identifier for the same principal in a particular format or federation namespace, it can send a request to  
 2671 the identity provider using this protocol.

2672 For example, a service provider that wishes to communicate with another service provider with whom it  
 2673 does not share an identifier for the principal can use an identity provider that shares an identifier for the  
 2674 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,  
 2675 with which it can communicate with the second service provider.

2676 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a  
 2677 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

### 2678 3.8.1 Element <NameIDMappingRequest>

2679 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
 2680 <NameIDMappingRequest> message. This message has the complex type  
 2681 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2682 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2683 The identifier and associated descriptive data that specify the principal as currently recognized by the  
 2684 requester and the responder. (For more information on this element, see Section 2.2.)

2685 <NameIDPolicy> [Required]

2686 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2687 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
 2688 binding used to deliver the message.

2689 The following schema fragment defines the <NameIDMappingRequest> element and its  
 2690 **NameIDMappingRequestType** complex type:

```

2691 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2692 <complexType name="NameIDMappingRequestType">
2693   <complexContent>
2694     <extension base="samlp:RequestAbstractType">
2695       <sequence>
2696         <choice>
2697           <element ref="saml:BaseID"/>
2698           <element ref="saml:NameID"/>
2699           <element ref="saml:EncryptedID"/>
2700         </choice>
2701         <element ref="samlp:NameIDPolicy"/>
2702       </sequence>
2703     </extension>
2704   </complexContent>
2705 </complexType>
  
```

### 3.8.2 Element <NameIDMappingResponse>

The recipient of a <NameIDMappingRequest> message MUST respond with a <NameIDMappingResponse> message. This message has the complex type **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

<saml:NameID> or <saml:EncryptedID> [Required]

The identifier and associated attributes that specify the principal in the manner requested, usually in encrypted form. (For more information on this element, see Section 2.2.)

The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <NameIDMappingResponse> element and its **NameIDMappingResponseType** complex type:

```
<element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
<complexType name="NameIDMappingResponseType">
  <complexContent>
    <extension base="samlp:StatusResponseType">
      <choice>
        <element ref="saml:NameID"/>
        <element ref="saml:EncryptedID"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

### 3.8.3 Processing Rules

If the responder does not recognize the principal identified in the request, it MAY respond with an error <Status> containing a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

At the responder's discretion, the urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

All other processing rules associated with the underlying request and response messages MUST be observed.

## 4 SAML Versioning

The SAML specification set is versioned in two independent ways. Each is discussed in the following sections, along with processing rules for detecting and handling version differences. Also included are guidelines on when and why specific version information is expected to change in future revisions of the specification.

When version information is expressed as both a Major and Minor version, it is expressed in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

$(Major_B > Major_A) \text{ OR } ( (Major_B = Major_A) \text{ AND } (Minor_B > Minor_A) )$

### 4.1 SAML Specification Set Version

Each release of the SAML specification set will contain a major and minor version designation describing its relationship to earlier and later versions of the specification set. The version will be expressed in the content and filenames of published materials, including the specification set documents and XML schema documents. There are no normative processing rules surrounding specification set versioning, since it merely encompasses the collective release of normative specification documents which themselves contain processing rules.

The overall size and scope of changes to the specification set documents will informally dictate whether a set of changes constitutes a major or minor revision. In general, if the specification set is backwards compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major revision.

#### 4.1.1 Schema Version

As a non-normative documentation mechanism, any XML schema documents published as part of the specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the form *Major.Minor*, reflecting the specification set version in which it has been published. Validating implementations MAY use the attribute as a means of distinguishing which version of a schema is being used to validate messages, or to support multiple versions of the same logical schema.

#### 4.1.2 SAML Assertion Version

The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the assertions of the same version. That is, specification set version 1.0 describes assertion version 1.0, and so on.

There is explicitly NO relationship between the assertion version and the target XML namespace specified for the schema definitions for that assertion version.

The following processing rules apply:

- A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion version number not supported by the authority.
- A SAML relying party MUST NOT process any assertion with a major assertion version number not supported by the relying party.
- A SAML relying party MAY process or MAY reject an assertion whose minor assertion version number is higher than the minor assertion version number supported by the relying party. However, all assertions that share a major assertion version number MUST share the same general

2779 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For  
 2780 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the  
 2781 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the  
 2782 likely effects of schema evolution.)

### 2783 4.1.3 SAML Protocol Version

2784 The various SAML protocols' request and response elements contain an attribute for expressing the major  
 2785 and minor version of the request or response message using a string of the form *Major.Minor*. Each  
 2786 version of the SAML specification set will be construed so as to document the syntax, semantics, and  
 2787 processing rules of the protocol messages of the same version. That is, specification set version 1.0  
 2788 describes request and response version V1.0, and so on.

2789 There is explicitly NO relationship between the protocol version and the target XML namespace specified  
 2790 for the schema definitions for that protocol version.

2791 The version numbers used in SAML protocol request and response elements will match for any particular  
 2792 revision of the SAML specification set.

#### 2793 4.1.3.1 Request Version

2794 The following processing rules apply to requests:

- 2795 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
 2796 SAML requester and the SAML responder.
- 2797 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD  
 2798 assume that the responder supports requests with the highest request version supported by the  
 2799 requester.
- 2800 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version  
 2801 number matching a response version number that the requester does not support.
- 2802 • A SAML responder MUST reject any request with a major request version number not supported by  
 2803 the responder.
- 2804 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
 2805 higher than the highest supported request version that it supports. However, all requests that share  
 2806 a major request version number MUST share the same general processing rules and semantics,  
 2807 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the  
 2808 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill  
 2809 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

#### 2810 4.1.3.2 Response Version

2811 The following processing rules apply to responses:

- 2812 • A SAML responder MUST NOT issue a response message with a response version number higher  
 2813 than the request version number of the corresponding request message.
- 2814 • A SAML responder MUST NOT issue a response message with a major response version number  
 2815 lower than the major request version number of the corresponding request message except to  
 2816 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2817 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a  
 2818 top-level `<StatusCode>` value of  
 2819 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting  
 2820 one of the following second-level values:



2821 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,  
 2822 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or  
 2823 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

### 2824 4.1.3.3 Permissible Version Combinations

2825 Assertions of a particular major version appear only in response messages of the same major version, as  
 2826 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For  
 2827 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1  
 2828 response message, if the appropriate assertion schema is referenced during namespace importation. But  
 2829 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major  
 2830 versions.

## 2831 4.2 SAML Namespace Version

2832 XML schema documents published as part of the specification set contain one or more target  
 2833 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct  
 2834 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that  
 2835 part of the specification.

2836 The namespace URI references defined by the specification set will generally contain version information  
 2837 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond  
 2838 to the major and minor version of the specification set in which the namespace is first introduced and  
 2839 defined. This information is not typically consumed by an XML processor, which treats the namespace  
 2840 opaquely, but is intended to communicate the relationship between the specification set and the  
 2841 namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are  
 2842 listed in Section 8.

2843 As a general rule, implementers can expect the namespaces and the associated schema definitions  
 2844 defined by a major revision of the specification set to remain valid and stable across minor revisions of the  
 2845 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but  
 2846 this is expected to be rare. In such cases, the older namespaces and their associated definitions should  
 2847 be expected to remain valid until a major specification set revision.

### 2848 4.2.1 Schema Evolution

2849 In general, maintaining namespace stability while adding or changing the content of a schema are  
 2850 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
 2851 older implementations will react to any given change, making forward compatibility difficult to achieve.  
 2852 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace  
 2853 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),  
 2854 implementations should expect forward-compatible schema changes in minor revisions, allowing new  
 2855 messages to validate against older schemas.

2856 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
 2857 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
 2858 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such  
 2859 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples  
 2860 include new query, statement, or condition types.

## 5 SAML and XML Signature Syntax and Processing

SAML assertions and SAML protocol request and response messages may be signed, with the following benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the message originator supports message integrity, authentication of message origin to a destination, and, if the signature is based on the originator's public-private key pair, non-repudiation of origin.

A digital signature is not always required in SAML. For example, in some circumstances, signatures may be "inherited," such as when an unsigned assertion gains protection from a signature on the containing protocol response message. "Inherited" signatures should be used with care when the contained object (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context must be retained to allow validation, exposing the XML content and adding potentially unnecessary overhead. As another example, the SAML relying party or SAML requester may have obtained an assertion or protocol message from the SAML asserting party or SAML responder directly (with no intermediaries) through a secure channel, with the asserting party or SAML responder having authenticated to the relying party or SAML responder by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based mechanisms, and so on. In addition, the applicable security requirements depend on the communicating applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all other contexts, digital signatures be used for assertions and request and response messages. Specifically:

- A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting party SHOULD be signed by the SAML asserting party.
- A SAML protocol message arriving at a destination from an entity other than the originating sender SHOULD be signed by the sender.
- Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are intended to be the primary SAML signature mechanism, but this specification attempts to ensure compatibility with profiles that may require other mechanisms.
- Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be enveloped.

### 5.1 Signing Assertions

All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as described in Section 2.

### 5.2 Request/Response Signing

All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected in the schema as described in Section 3.

### 5.3 Signature Inheritance

A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>` or a request or response, which may be signed. When a SAML assertion does not contain a `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,

2904 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
 2905 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
 2906 and signature options.

2907 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
 2908 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY  
 2909 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
 2910 information from the surrounding context, but no such inheritance should be inferred unless specifically  
 2911 identified by the profile.

## 2912 **5.4 XML Signature Profile**

2913 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
 2914 and many choices. This section details constraints on these facilities so that SAML processors do not  
 2915 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
 2916 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**  
 2917 attribute on `<Assertion>` and the various request and response elements. These attributes are  
 2918 collectively referred to in this section as the identifier attributes.

2919 Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML  
 2920 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to  
 2921 SAML content are free to define other approaches.

### 2922 **5.4.1 Signing Formats and Algorithms**

2923 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
 2924 detached.

2925 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
 2926 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
 2927 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2928 **5.4.2 References**

2929 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of  
 2930 the assertion or protocol message being signed. The assertion's or protocol message's root element may  
 2931 or may not be the root element of the actual XML document containing the signed assertion or protocol  
 2932 message (e.g., it might be contained within a SOAP envelope).

2933 Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the **ID**  
 2934 attribute value of the root element of the assertion or protocol message being signed. For example, if the  
 2935 **ID** attribute value is "foo", then the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

### 2936 **5.4.3 Canonicalization Method**

2937 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,  
 2938 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a  
 2939 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over  
 2940 SAML messages embedded in an XML context can be verified independent of that context.

### 2941 **5.4.4 Transforms**

2942 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
 2943 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive

2944 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2945 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2946 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2947 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can  
2948 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2949 applying the transforms manually to the content and reverifying the result as consisting of the same SAML  
2950 message.

## 2951 5.4.5 KeyInfo

2952 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of  
2953 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be  
2954 absent.

## 2955 5.4.6 Example

2956 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2957 added for readability; the signatures are not valid and cannot be successfully verified.

```
2958 <Response
2959   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
2960   ID="_c7055387-af61-4fce-8b98-e2927324b306"
2961   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
2962   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2963   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
2964   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2965     <ds:SignedInfo>
2966       <ds:CanonicalizationMethod
2967         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2968       <ds:SignatureMethod
2969         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2970       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
2971         <ds:Transforms>
2972           <ds:Transform
2973             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
2974 signature" />
2975           <ds:Transform
2976             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2977             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
2978               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2979           </ds:Transform>
2980         </ds:Transforms>
2981         <ds:DigestMethod
2982           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2983         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2984       </ds:Reference>
2985     </ds:SignedInfo>
2986     <ds:SignatureValue>
2987       x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
2988       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWaptaK1ywS7gFgsD01qjyen3CP+m3D
2989       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
2990     </ds:SignatureValue>
2991     <ds:KeyInfo>
2992       <ds:X509Data>
2993         <ds:X509Certificate>
2994           MIIICyJCCAjoGAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTA1VT
2995           MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2996           FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2lvbiBvZiBJ
2997           bmZvcmlhdGlvb2ZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgc0Eg
2998           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTE2MDkwNDA3Mjc1MVoYsxCzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
2999           CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
```

```

3000 Ym9yMQ4wDAYDVQKKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3001 dTEncMCUGCSqGSIB3DQeJARYYcm9vdEBzaGlms5pbnRlcm5ldDIuZWRLMIGfMA0G
3002 CSqGSIB3DQeBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
3003 IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYiE7x4XAWIrCP+
3004 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W027rhRjE
3005 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3006 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3007 qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3008 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
3009 </ds:X509Certificate>
3010 </ds:X509Data>
3011 </ds:KeyInfo>
3012 </ds:Signature>
3013 <Status>
3014   <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3015 </Status>
3016 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3017   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3018   xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3019   <Issuer>https://www.opensaml.org/IDP</Issuer>
3020   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3021     <ds:SignedInfo>
3022       <ds:CanonicalizationMethod
3023         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3024       <ds:SignatureMethod
3025         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3026       <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3027         <ds:Transforms>
3028           <ds:Transform
3029             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3030 signature"/>
3031           <ds:Transform
3032             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3033             <InclusiveNamespaces
3034               PrefixList="#default saml ds xs xsi"
3035               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#">
3036             </ds:Transform>
3037           </ds:Transforms>
3038           <ds:DigestMethod
3039             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3040           <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3041         </ds:Reference>
3042       </ds:SignedInfo>
3043       <ds:SignatureValue>
3044         hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3045         7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtP3TD
3046         Mwul/cBUj2OtBZQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3047       </ds:SignatureValue>
3048     </ds:KeyInfo>
3049     <ds:X509Data>
3050       <ds:X509Certificate>
3051       MIIcYjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaksCzAJBgNVBAYTA1VT
3052       MRIwEAYDVQQIEw1xXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
3053       FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLIEYjEaXZpc2l2b2V3ZiBJ
3054       bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3055       LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVVoXDTA2MDkwNDA3Mjc1MVowgYys
3056       CzAJBgNVBAYTA1VTREwDwYDVQQIEwhNaWNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
3057       Ym9yMQ4wDAYDVQKKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJuZXQyLmVk
3058       dTEncMCUGCSqGSIB3DQeJARYYcm9vdEBzaGlms5pbnRlcm5ldDIuZWRLMIGfMA0G
3059       CSqGSIB3DQeBAQUAA4GNADCBiQKBgQDZSAb2sxvhAXnXVIVTx8vuRay+x50z7GJj
3060       IHRYQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYiE7x4XAWIrCP+
3061       c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W027rhRjE
3062       pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3063       hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3064       qgi7lFV6MDkhmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3065       8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==

```

```
3066         </ds:X509Certificate>
3067     </ds:X509Data>
3068     </ds:KeyInfo>
3069 </ds:Signature>
3070 <Subject>
3071     <NameID
3072         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3073         scott@example.org
3074     </NameID>
3075     <SubjectConfirmation
3076         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3077 </Subject>
3078 <Conditions NotBefore="2003-04-17T00:46:02Z"
3079     NotOnOrAfter="2003-04-17T00:51:02Z">
3080     <AudienceRestriction>
3081         <Audience>http://www.opensaml.org/SP</Audience>
3082     </AudienceRestriction>
3083 </Conditions>
3084 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3085     <AuthnContext>
3086         <AuthnContextClassRef>
3087             urn:oasis:names:tc:SAML:2.0:ac:classes:Password
3088         </AuthnContextClassRef>
3089     </AuthnContext>
3090 </AuthnStatement>
3091 </Assertion>
3092 </Response>
```



## 6 SAML and XML Encryption Syntax and Processing

Encryption is used as the means to implement confidentiality. The most common motives for confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness of some other security mechanism. For example, a secret password or key may be encrypted.

Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- Communications confidentiality may be provided by mechanisms associated with a particular binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>` element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.
- An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

### 6.1 General Considerations

Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace the plaintext information in the same location within the XML instance. The `<EncryptedData>` element's `Type` attribute SHOULD be used and, if it is present, MUST have the value `http://www.w3.org/2001/04/xmlenc#Element`.

Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

### 6.2 Combining Signatures and Encryption

Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in the reverse order that signing and encryption were performed.

- When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and placed within the `<Assertion>` element before the element is encrypted.
- When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be performed first and then the signature calculated over the assertion or message containing the encrypted element.

## 7 SAML Extensibility

SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas. An example of an application that extends SAML assertions is the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions and protocols.

See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can be combined with extensions to put the SAML framework to new uses.

### 7.1 Schema Extension

Note that elements in the SAML schemas are blocked from substitution, which means that no SAML elements can serve as the head element of a substitution group. However, SAML types are not defined as *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that extensions are typically defined only as types rather than elements, and are included in SAML instances by means of an `xsi:type` attribute.

The following sections discuss only elements and types that have been specifically designed to support extensibility.

#### 7.1.1 Assertion Schema Extension

The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the assertion package and the statements it contains, if the extension mechanism is used for either part.

The following elements are intended specifically for use as extension points in an extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived type:

- `<BaseID>` and **BaseIDAbstractType**
- `<Condition>` and **ConditionAbstractType**
- `<Statement>` and **StatementAbstractType**
- The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:
  - `<AuthnStatement>` and **AuthnStatementType**
  - `<AttributeStatement>` and **AttributeStatementType**
  - `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
  - `<AudienceRestriction>` and **AudienceRestrictionType**
  - `<ProxyRestriction>` and **ProxyRestrictionType**
  - `<OneTimeUse>` and **OneTimeUseType**

#### 7.1.2 Protocol Schema Extension

The following SAML protocol elements are intended specifically for use as extension points in an extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived type:

- `<Request>` and **RequestAbstractType**
- `<SubjectQuery>` and **SubjectQueryAbstractType**

The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:

- `<AuthnQuery>` and **AuthnQueryType**
- `<AuthzDecisionQuery>` and **AuthzDecisionQueryType**
- `<AttributeQuery>` and **AttributeQueryType**
- **StatusResponseType**

## 7.2 Schema Wildcard Extension Points

The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes from arbitrary namespaces, which serves as a built-in extension point without requiring an extension schema.

### 7.2.1 Assertion Extension Points

The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- `<SubjectConfirmationData>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<AuthnContextDecl>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<AttributeValue>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- `<Advice>` and **AdviceType**: In addition to SAML-native elements, allows elements from other namespaces with lax schema validation processing.

The following constructs in the assertion schema allow arbitrary global attributes:

- `<Attribute>` and **AttributeType**

### 7.2.2 Protocol Extension Points

The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- `<Extensions>` and **ExtensionsType**: Allows elements from other namespaces with lax schema validation processing.
- `<StatusDetail>` and **StatusDetailType**: Allows elements from other namespaces with lax schema validation processing.
- `<ArtifactResponse>` and **ArtifactResponseType**: Allows elements from any namespaces with lax schema validation processing. (It is specifically intended to carry a SAML request or response message element, however.)

## 7.3 Identifier Extension

SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8. However, it is always possible to define additional URI-based identifiers for these purposes. It is RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should the meaning of a given URI used as such an identifier significantly change, or be used to mean two different things.

## 8 SAML-Defined Identifiers

The following sections define URI-based identifiers for common resource access actions, subject name identifier formats, and attribute name formats.

Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of the most current RFC that specifies the protocol is used. URI references created specifically for SAML have one of the following stems, according to the specification set version in which they were first introduced:

```
urn:oasis:names:tc:SAML:1.0:
urn:oasis:names:tc:SAML:1.1:
urn:oasis:names:tc:SAML:2.0:
```

### 8.1 Action Namespace Identifiers

The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to common sets of actions to perform on resources.

#### 8.1.1 Read/Write/Execute/Delete/Control

**URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

Defined actions:

`Read Write Execute Delete Control`

These actions are interpreted as follows:

**Read**

The subject may read the resource.

**Write**

The subject may modify the resource.

**Execute**

The subject may execute the resource.

**Delete**

The subject may delete the resource.

**Control**

The subject may specify the access control policy for the resource.

#### 8.1.2 Read/Write/Execute/Delete/Control with Negation

**URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

Defined actions:

`Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action `~Read` is affirmatively denied read permission.

3234 A SAML authority MUST NOT authorize both an action and its negated form.

### 3235 8.1.3 Get/Head/Put/Post

3236 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3237 Defined actions:

3238 GET HEAD PUT POST

3239 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
3240 the GET action on a resource is authorized to retrieve it.

3241 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST  
3242 actions to the write permission. The correspondence is not exact however since an HTTP GET operation  
3243 may cause data to be modified and a POST operation may cause modification to a resource other than  
3244 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

### 3245 8.1.4 UNIX File Permissions

3246 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3247 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3248 The action string is a four-digit numeric code:

3249 *extended user group world*

3250 Where the *extended* access permission has the value

3251 +2 if sgid is set

3252 +4 if suid is set

3253 The *user group* and *world* access permissions have the value

3254 +1 if execute permission is granted

3255 +2 if write permission is granted

3256 +4 if read permission is granted

3257 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read  
3258 and execute; and world read.

## 3259 8.2 Attribute Name Format Identifiers

3260 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex  
3261 type to refer to the classification of the attribute name for purposes of interpreting the name.

### 3262 8.2.1 Unspecified

3263 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3264 The interpretation of the attribute name is left to individual implementations.

## 8.2.2 URI Reference

**URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-specific. See [SAMLProf] for attribute profiles that make use of this identifier.

## 8.2.3 Basic

**URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to the primitive type **xs:Name** as defined in [Schema2] Section 3.3.6. See [SAMLProf] for attribute profiles that make use of this identifier.

## 8.3 Name Identifier Format Identifiers

The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the associated processing rules, if any.

**Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for SAML V2.0.

### 8.3.1 Unspecified

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

The interpretation of the content of the element is left to individual implementations.

### 8.3.2 Email Address

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] Section 3.4.1. An addr-spec has the form local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

### 8.3.3 X.509 Subject Name

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

Indicates that the content of the element is in the form specified for the contents of the `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors should note that the XML Signature specification specifies encoding rules for X.509 subject names that differ from the rules given in IETF RFC 2253 [RFC 2253].

### 8.3.4 Windows Domain Qualified Name

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName



3298 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
 3299 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator  
 3300 MAY be omitted.

### 3301 **8.3.5 Kerberos Principal Name**

3302 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3303 Indicates that the content of the element is in the form of a Kerberos principal name using the format  
 3304 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and  
 3305 realm are described in IETF RFC 1510 [RFC 1510].

### 3306 **8.3.6 Entity Identifier**

3307 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3308 Indicates that the content of the element is the identifier of an entity that provides SAML-based services  
 3309 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service  
 3310 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to  
 3311 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make  
 3312 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be  
 3313 used in other elements and attributes whose purpose is to identify a system entity in various protocol  
 3314 exchanges.

3315 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is  
 3316 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3317 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

### 3318 **8.3.7 Persistent Identifier**

3319 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3320 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to  
 3321 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers  
 3322 generated by identity providers MUST be constructed using pseudo-random values that have no  
 3323 discernible correspondence with the subject's actual identifier (for example, username). The intent is to  
 3324 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.  
 3325 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3326 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity  
 3327 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived  
 3328 from the context of the message containing the element, such as the issuer of a protocol message or an  
 3329 assertion containing the identifier in its subject. Note that a different system entity might later issue its own  
 3330 protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in  
 3331 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be  
 3332 omitted in such a case).

3333 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service  
 3334 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be  
 3335 omitted if the element is contained in a message intended only for consumption directly by the service  
 3336 provider, and the value would be the unique identifier of that service provider.

3337 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most  
 3338 recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been  
 3339 established, then the attribute MUST be omitted.

Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared in clear text with providers other than the providers that have established the shared identifier. Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and protections. Deployments without such requirements are free to use other kinds of identifiers in their SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

Note also that while persistent identifiers are typically used to reflect an account linking relationship between a pair of providers, a service provider is not obligated to recognize or make use of the long term nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly different and does not affect the behavior of the identity provider or any processing rules specific to persistent identifiers in the protocols defined in this specification.

Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of creation, but not of use. If a persistent identifier is created by a particular identity provider, the `NameQualifier` attribute value is permanently established at that time. If a service provider that receives such an identifier takes on the role of an identity provider and issues its own assertion containing that identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It might alternatively choose to create its own persistent identifier to represent the principal and link the two values. This is a deployment decision.

### 8.3.8 Transient Identifier

**URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of 256 characters.

The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in accordance with the rules specified in Section 8.3.7.

## 8.4 Consent Identifiers

The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and **StatusResponseType** complex types to communicate whether a principal gave consent, and under what conditions, for the message.

### 8.4.1 Unspecified

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

No claim as to principal consent is being made.

### 8.4.2 Obtained

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

Indicates that a principal's consent has been obtained by the issuer of the message.

### 8.4.3 Prior

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`

3378 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to  
3379 the action that initiated the message.

#### 3380 **8.4.4 Implicit**

3381 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3382 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the  
3383 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically  
3384 more proximal to the action in time and presentation than prior consent, such as part of a session of  
3385 activities.

#### 3386 **8.4.5 Explicit**

3387 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3388 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the  
3389 action that initiated the message.

#### 3390 **8.4.6 Unavailable**

3391 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3392 Indicates that the issuer of the message did not obtain consent.

#### 3393 **8.4.7 Inapplicable**

3394 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3395 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

## 9 References

The following works are cited in the body of this specification.

### 9.1 Normative References

- [Excl-C14N]** J. Boyer et al. *Exclusive XML Canonicalization Version 1.0*. World Wide Web Consortium, July 2002. See <http://www.w3.org/TR/xml-exc-c14n/>.
- [Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- [Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-2/>.
- [XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. See <http://www.w3.org/TR/REC-xml>.
- [XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>. Note that this specification normatively references [XMLEnc-XSD], listed below.
- [XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- [XMLNS]** T. Bray et al. *Namespaces in XML*. World Wide Web Consortium, January 1999. See <http://www.w3.org/TR/REC-xml-names>.
- [XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web Consortium, February 2002. See <http://www.w3.org/TR/xmldsig-core/>. Note that this specification normatively references [XMLSig-XSD], listed below.
- [XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd>.

### 9.2 Non-Normative References

- [LibertyProt]** J. Beatty et al. *Liberty Protocols and Schema Specification Version 1.1*. Liberty Alliance Project, January 2003. See [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- [RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. See <http://www.ietf.org/rfc/rfc1510.txt>.
- [RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- [RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See <http://www.ietf.org/rfc/rfc2246.txt>.
- [RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. See <http://www.ietf.org/rfc/rfc2253.txt>.
- [RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. See <http://www.ietf.org/rfc/rfc2396.txt>.
- [RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. See <http://www.ietf.org/rfc/rfc2822.txt>.

3439	<b>[RFC 3075]</b>	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. See <a href="http://www.ietf.org/rfc/rfc3075.txt">http://www.ietf.org/rfc/rfc3075.txt</a> .
3440		
3441	<b>[RFC 3513]</b>	R. Hinden, S. Deering, <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. See <a href="http://www.ietf.org/rfc/rfc3513.txt">http://www.ietf.org/rfc/rfc3513.txt</a> .
3442		
3443	<b>[SAMLAuthnCxt]</b>	J. Kemp et al. <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3444		
3445		
3446	<b>[SAMLBind]</b>	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3447		
3448		
3449	<b>[SAMLConform]</b>	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3450		
3451		
3452	<b>[SAMLGloss]</b>	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3453		
3454		
3455	<b>[SAMLMeta]</b>	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3456		
3457		
3458	<b>[SAML P-XSD]</b>	S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3459		
3460		
3461	<b>[SAMLProf]</b>	S. Cantor et al. <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3462		
3463		
3464	<b>[SAMLSecure]</b>	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3465		
3466		
3467		
3468	<b>[SAML TechOvw]</b>	J. Hughes et al. SAML Technical Overview. OASIS, February 2005. Document ID sstc-saml-tech-overview-2.0-draft-03. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3469		
3470		
3471	<b>[SAML-XSD]</b>	S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3472		
3473		
3474	<b>[SSL3]</b>	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
3475		
3476	<b>[UNICODE-C]</b>	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. See <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a> .
3477		
3478	<b>[W3C-CHAR]</b>	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. See <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a> .
3479		
3480	<b>[W3C-CharMod]</b>	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. See <a href="http://www.w3.org/TR/charmod-norm/">http://www.w3.org/TR/charmod-norm/</a> .
3481		
3482		
3483	<b>[XACML]</b>	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See <a href="http://www.oasis-open.org/committees/xacml">http://www.oasis-open.org/committees/xacml</a> .
3484		
3485	<b>[XML-ID]</b>	J. Marsh et al. <i>xml:id Version 1.0</i> , World Wide Web Consortium, April 2004. See <a href="http://www.w3.org/TR/xml-id/">http://www.w3.org/TR/xml-id/</a> .
3486		

## Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- Conor Cahill, AOL
- John Hughes, Atos Origin
- Hal Lockhart, BEA Systems
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiester, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems



- 3529 • Eve Maler, Sun Microsystems
- 3530 • Ronald Monzillo, Sun Microsystems
- 3531 • Emily Xu, Sun Microsystems
- 3532 • Greg Whitehead, Trustgenix

3533

3534 The editors also would like to acknowledge the following former SSTC members for their contributions to  
 3535 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 3536 • Stephen Farrell, Baltimore Technologies
- 3537 • David Orchard, BEA Systems
- 3538 • Krishna Sankar, Cisco Systems
- 3539 • Zahid Ahmed, CommerceOne
- 3540 • Tim Alsop, CyberSafe Limited
- 3541 • Carlisle Adams, Entrust
- 3542 • Tim Moses, Entrust
- 3543 • Nigel Edwards, Hewlett-Packard
- 3544 • Joe Pato, Hewlett-Packard
- 3545 • Bob Blakley, IBM
- 3546 • Marlena Erdos, IBM
- 3547 • Marc Chanliau, Netegrity
- 3548 • Chris McLaren, Netegrity
- 3549 • Lynne Rosenthal, NIST
- 3550 • Mark Skall, NIST
- 3551 • Charles Knouse, Oblix
- 3552 • Simon Godik, Overxeer
- 3553 • Charles Norwood, SAIC
- 3554 • Evan Prodromou, Securant
- 3555 • Robert Griffin, RSA Security (former editor)
- 3556 • Sai Allarvarpu, Sun Microsystems
- 3557 • Gary Ellison, Sun Microsystems
- 3558 • Chris Ferris, Sun Microsystems
- 3559 • Mike Myers, Traceroute Security
- 3560 • Phillip Hallam-Baker, VeriSign (former editor)
- 3561 • James Vanderbeek, Vodafone
- 3562 • Mark O'Neill, Vordel
- 3563 • Tony Palmer, Vordel

3564

3565 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
 3566 input to the OASIS Security Assertions Markup Language specifications:

- 3567 • Thomas Gross, IBM
- 3568 • Birgit Pfitzmann, IBM

## Appendix B. Notices

3569

3570 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
 3571 might be claimed to pertain to the implementation or use of the technology described in this document or  
 3572 the extent to which any license under such rights might or might not be available; neither does it represent  
 3573 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
 3574 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
 3575 available for publication and any assurances of licenses to be made available, or the result of an attempt  
 3576 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
 3577 users of this specification, can be obtained from the OASIS Executive Director.

3578 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
 3579 other proprietary rights which may cover technology that may be required to implement this specification.  
 3580 Please address the information to the OASIS Executive Director.

3581 **Copyright © OASIS Open 2005. All Rights Reserved.**

3582 This document and translations of it may be copied and furnished to others, and derivative works that  
 3583 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
 3584 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
 3585 this paragraph are included on all such copies and derivative works. However, this document itself may  
 3586 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
 3587 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
 3588 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
 3589 into languages other than English.

3590 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
 3591 or assigns.

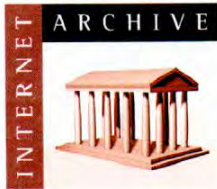
3592 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
 3593 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
 3594 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
 3595 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

# EXHIBIT 9



# EXHIBIT 10





www.archive.org  
415.561.6767  
415.840-0391 e-fax

Internet Archive  
300 Funston Avenue  
San Francisco, CA 94118

## AFFIDAVIT OF CHRISTOPHER BUTLER

1. I am the Office Manager at the Internet Archive, located in San Francisco, California. I make this declaration of my own personal knowledge.

2. The Internet Archive is a website that provides access to a digital library of Internet sites and other cultural artifacts in digital form. Like a paper library, we provide free access to researchers, historians, scholars, and the general public. The Internet Archive has partnered with and receives support from various institutions, including the Library of Congress.

3. The Internet Archive has created a service known as the Wayback Machine. The Wayback Machine makes it possible to surf more than 450 billion pages stored in the Internet Archive's web archive. Visitors to the Wayback Machine can search archives by URL (i.e., a website address). If archived records for a URL are available, the visitor will be presented with a list of available dates. The visitor may select one of those dates, and then begin surfing on an archived version of the Web. The links on the archived files, when served by the Wayback Machine, point to other archived files (whether HTML pages or images). If a visitor clicks on a link on an archived page, the Wayback Machine will serve the archived file with the closest available date to the page upon which the link appeared and was clicked.

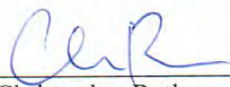
4. The archived data made viewable and browseable by the Wayback Machine is compiled using software programs known as crawlers, which surf the Web and automatically store copies of web files, preserving these files as they exist at the point of time of capture.

5. The Internet Archive assigns a URL on its site to the archived files in the format `http://web.archive.org/web/[Year in yyyy][Month in mm][Day in dd][Time code in hh:mm:ss]/[Archived URL]`. Thus, the Internet Archive URL `http://web.archive.org/web/19970126045828/http://www.archive.org/` would be the URL for the record of the Internet Archive home page HTML file (`http://www.archive.org/`) archived on January 26, 1997 at 4:58 a.m. and 28 seconds (1997/01/26 at 04:58:28). A web browser may be set such that a printout from it will display the URL of a web page in the printout's footer. The date assigned by the Internet Archive applies to the HTML file but not to image files linked therein. Thus images that appear on a page may not have been archived on the same date as the HTML file. Likewise, if a website is designed with "frames," the date assigned by the Internet Archive applies to the frameset as a whole, and not the individual pages within each frame.

6. Attached hereto as Exhibit A are true and accurate copies of printouts of the Internet Archive's records of the HTML files or PDF files for the URLs and the dates specified in the footer of the printout (HTML) or attached coversheet (PDF).

7. I declare under penalty of perjury that the foregoing is true and correct.

DATE: 1/26/17

  
\_\_\_\_\_  
Christopher Butler



CALIFORNIA JURAT

---

See Attached Document.

A notary public or other officer completing this certificate verifies only the identity of the individual who signed the document to which this certificate is attached, and not the truthfulness, accuracy, or validity of that document.

State of California  
County of San Francisco

Subscribed and sworn to (or affirmed) before me on this

26<sup>th</sup> day of January, 2017, by

Christopher Butler,

proved to me on the basis of satisfactory evidence to be the person who appeared before me.



Signature: Laurel Karr

# Exhibit A

# Index of /security/saml/v2.0

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
<a href="#">Parent Directory</a>	12-Sep-2005 14:26	-	
<a href="#">saml-2.0-os-xsd.zip</a>	12-Sep-2005 14:31	38k	
<a href="#">saml-2.0-os.zip</a>	17-Mar-2005 13:29	2.7M	
<a href="#">saml-authn-context-2.0-os.pdf</a>	17-Mar-2005 13:29	346k	
<a href="#">saml-bindings-2.0-os.pdf</a>	17-Mar-2005 13:29	431k	
<a href="#">saml-conformance-2.0-os.pdf</a>	17-Mar-2005 13:29	170k	
<a href="#">saml-core-2.0-os.pdf</a>	17-Mar-2005 13:29	616k	
<a href="#">saml-glossary-2.0-os.pdf</a>	17-Mar-2005 13:29	189k	
<a href="#">saml-metadata-2.0-os.pdf</a>	17-Mar-2005 13:29	314k	
<a href="#">saml-profiles-2.0-os.pdf</a>	17-Mar-2005 13:29	542k	
<a href="#">saml-schema-assertion-2.0.xsd</a>	12-Sep-2005 14:31	13k	
<a href="#">saml-schema-authn-context-2.0.xsd</a>	12-Sep-2005 14:31	1k	
<a href="#">saml-schema-authn-context-auth-telephony-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-ip-2.0.xsd</a>	12-Sep-2005 14:31	2k	
<a href="#">saml-schema-authn-context-ippword-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-kerberos-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-mobileonefactor-reg-2.0.xsd</a>	12-Sep-2005 14:31	7k	
<a href="#">saml-schema-authn-context-mobileonefactor-unreg-2.0.xsd</a>	12-Sep-2005 14:31	7k	
<a href="#">saml-schema-authn-context-mobiletwofactor-reg-2.0.xsd</a>	12-Sep-2005 14:31	8k	
<a href="#">saml-schema-authn-context-mobiletwofactor-unreg-2.0.xsd</a>	12-Sep-2005 14:31	8k	
<a href="#">saml-schema-authn-context-nomad-telephony-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-personal-telephony-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-pgp-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-ppt-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-pword-2.0.xsd</a>	12-Sep-2005 14:31	2k	
<a href="#">saml-schema-authn-context-session-2.0.xsd</a>	12-Sep-2005 14:31	2k	
<a href="#">saml-schema-authn-context-smartcard-2.0.xsd</a>	12-Sep-2005 14:31	2k	
<a href="#">saml-schema-authn-context-smartcardpki-2.0.xsd</a>	12-Sep-2005 14:31	5k	
<a href="#">saml-schema-authn-context-softwarepki-2.0.xsd</a>	12-Sep-2005 14:31	5k	
<a href="#">saml-schema-authn-context-spki-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-srp-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-sslcert-2.0.xsd</a>	12-Sep-2005 14:31	4k	
<a href="#">saml-schema-authn-context-telephony-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-timesync-2.0.xsd</a>	12-Sep-2005 14:31	4k	
<a href="#">saml-schema-authn-context-types-2.0.xsd</a>	12-Sep-2005 14:31	29k	
<a href="#">saml-schema-authn-context-x509-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-authn-context-xmldsig-2.0.xsd</a>	12-Sep-2005 14:31	3k	
<a href="#">saml-schema-dce-2.0.xsd</a>	12-Sep-2005 14:31	1k	
<a href="#">saml-schema-ecp-2.0.xsd</a>	12-Sep-2005 14:31	2k	
<a href="#">saml-schema-metadata-2.0.xsd</a>	12-Sep-2005 14:31	16k	
<a href="#">saml-schema-protocol-2.0.xsd</a>	12-Sep-2005 14:31	13k	
<a href="#">saml-schema-x500-2.0.xsd</a>	12-Sep-2005 14:31	1k	
<a href="#">saml-schema-xacml-2.0.xsd</a>	12-Sep-2005 14:31	1k	
<a href="#">saml-sec-consider-2.0-os.pdf</a>	17-Mar-2005 13:29	276k	

<https://web.archive.org/web/20060517032319/http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>



# Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0

OASIS Standard, 15 March 2005

## Document identifier:

saml-core-2.0-os

## Location:

<http://docs.oasis-open.org/security/saml/v2.0/>

## Editors:

Scott Cantor, Internet2  
John Kemp, Nokia  
Rob Philpott, RSA Security  
Eve Maler, Sun Microsystems

## SAML V2.0 Contributors:

Conor P. Cahill, AOL  
John Hughes, Atos Origin  
Hal Lockhart, BEA Systems  
Michael Beach, Boeing  
Rebekah Metz, Booz Allen Hamilton  
Rick Randall, Booz Allen Hamilton  
Thomas Wisniewski, Entrust  
Irving Reid, Hewlett-Packard  
Paula Austel, IBM  
Maryann Hondo, IBM  
Michael McIntosh, IBM  
Tony Nadalin, IBM  
Nick Ragouzis, Individual  
Scott Cantor, Internet2  
RL 'Bob' Morgan, Internet2  
Peter C Davis, Neustar  
Jeff Hodges, Neustar  
Frederick Hirsch, Nokia  
John Kemp, Nokia  
Paul Madsen, NTT  
Steve Anderson, OpenNetwork  
Prateek Mishra, Principal Identity  
John Linn, RSA Security  
Rob Philpott, RSA Security  
Jahan Moreh, Sigaba  
Anne Anderson, Sun Microsystems

Eve Maler, Sun Microsystems  
Ron Monzillo, Sun Microsystems  
Greg Whitehead, Trustgenix

**Abstract:**

This specification defines the syntax and semantics for XML-encoded assertions about authentication, attributes, and authorization, and for the protocols that convey this information.

**Status:**

This is an **OASIS Standard** document produced by the Security Services Technical Committee. It was approved by the OASIS membership on 1 March 2005.

Committee members should submit comments and potential errata to the [security-services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should submit them by filling out the web form located at [http://www.oasis-open.org/committees/comments/form.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=security). The committee will publish on its web page (<http://www.oasis-open.org/committees/security>) a catalog of any changes made to this document as a result of comments.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the Security Services TC (<http://www.oasis-open.org/committees/security/ipr.php>).



# Table of Contents

61	1 Introduction.....	7
62	1.1 Notation.....	7
63	1.2 Schema Organization and Namespaces.....	8
64	1.3 Common Data Types.....	8
65	1.3.1 String Values.....	8
66	1.3.2 URI Values.....	9
67	1.3.3 Time Values.....	9
68	1.3.4 ID and ID Reference Values.....	9
69	2 SAML Assertions.....	11
70	2.1 Schema Header and Namespace Declarations.....	11
71	2.2 Name Identifiers.....	12
72	2.2.1 Element <BaseID>.....	12
73	2.2.2 Complex Type NameIDType.....	13
74	2.2.3 Element <NameID>.....	14
75	2.2.4 Element <EncryptedID>.....	14
76	2.2.5 Element <Issuer>.....	15
77	2.3 Assertions.....	15
78	2.3.1 Element <AssertionIDRef>.....	15
79	2.3.2 Element <AssertionURIRef>.....	15
80	2.3.3 Element <Assertion>.....	15
81	2.3.4 Element <EncryptedAssertion>.....	17
82	2.4 Subjects.....	17
83	2.4.1 Element <Subject>.....	18
84	2.4.1.1 Element <SubjectConfirmation>.....	18
85	2.4.1.2 Element <SubjectConfirmationData>.....	19
86	2.4.1.3 Complex Type KeyInfoConfirmationDataType.....	20
87	2.4.1.4 Example of a Key-Confirmed <Subject>.....	21
88	2.5 Conditions.....	21
89	2.5.1 Element <Conditions>.....	21
90	2.5.1.1 General Processing Rules.....	22
91	2.5.1.2 Attributes NotBefore and NotOnOrAfter.....	23
92	2.5.1.3 Element <Condition>.....	23
93	2.5.1.4 Elements <AudienceRestriction> and <Audience>.....	23
94	2.5.1.5 Element <OneTimeUse>.....	24
95	2.5.1.6 Element <ProxyRestriction>.....	25
96	2.6 Advice.....	25
97	2.6.1 Element <Advice>.....	26
98	2.7 Statements.....	26
99	2.7.1 Element <Statement>.....	26
100	2.7.2 Element <AuthnStatement>.....	26
101	2.7.2.1 Element <SubjectLocality>.....	28
102	2.7.2.2 Element <AuthnContext>.....	28
103	2.7.3 Element <AttributeStatement>.....	29
104	2.7.3.1 Element <Attribute>.....	29
105	2.7.3.1.1 Element <AttributeValue>.....	30
106	2.7.3.2 Element <EncryptedAttribute>.....	31
107	2.7.4 Element <AuthzDecisionStatement>.....	31
108	2.7.4.1 Simple Type DecisionType.....	33
109	2.7.4.2 Element <Action>.....	33

110	2.7.4.3 Element <Evidence>.....	34
111	3 SAML Protocols.....	35
112	3.1 Schema Header and Namespace Declarations.....	35
113	3.2 Requests and Responses.....	36
114	3.2.1 Complex Type RequestAbstractType.....	36
115	3.2.2 Complex Type StatusResponseType.....	38
116	3.2.2.1 Element <Status>.....	39
117	3.2.2.2 Element <StatusCode>.....	39
118	3.2.2.3 Element <StatusMessage>.....	42
119	3.2.2.4 Element <StatusDetail>.....	42
120	3.3 Assertion Query and Request Protocol.....	42
121	3.3.1 Element <AssertionIDRequest>.....	42
122	3.3.2 Queries.....	42
123	3.3.2.1 Element <SubjectQuery>.....	42
124	3.3.2.2 Element <AuthnQuery>.....	43
125	3.3.2.2.1 Element <RequestedAuthnContext>.....	44
126	3.3.2.3 Element <AttributeQuery>.....	45
127	3.3.2.4 Element <AuthzDecisionQuery>.....	46
128	3.3.3 Element <Response>.....	46
129	3.3.4 Processing Rules.....	47
130	3.4 Authentication Request Protocol.....	47
131	3.4.1 Element <AuthnRequest>.....	48
132	3.4.1.1 Element <NameIDPolicy>.....	50
133	3.4.1.2 Element <Scoping>.....	51
134	3.4.1.3 Element <IDPList>.....	52
135	3.4.1.3.1 Element <IDPEntry>.....	52
136	3.4.1.4 Processing Rules.....	53
137	3.4.1.5 Proxying.....	54
138	3.4.1.5.1 Proxying Processing Rules.....	54
139	3.5 Artifact Resolution Protocol.....	55
140	3.5.1 Element <ArtifactResolve>.....	56
141	3.5.2 Element <ArtifactResponse>.....	56
142	3.5.3 Processing Rules.....	56
143	3.6 Name Identifier Management Protocol.....	57
144	3.6.1 Element <ManageNameIDRequest>.....	57
145	3.6.2 Element <ManageNameIDResponse>.....	58
146	3.6.3 Processing Rules.....	58
147	3.7 Single Logout Protocol.....	59
148	3.7.1 Element <LogoutRequest>.....	60
149	3.7.2 Element <LogoutResponse>.....	60
150	3.7.3 Processing Rules.....	61
151	3.7.3.1 Session Participant Rules.....	61
152	3.7.3.2 Session Authority Rules.....	62
153	3.8 Name Identifier Mapping Protocol.....	63
154	3.8.1 Element <NameIDMappingRequest>.....	63
155	3.8.2 Element <NameIDMappingResponse>.....	64
156	3.8.3 Processing Rules.....	64
157	4 SAML Versioning.....	65
158	4.1 SAML Specification Set Version.....	65
159	4.1.1 Schema Version.....	65
160	4.1.2 SAML Assertion Version.....	65
161	4.1.3 SAML Protocol Version.....	66
162	4.1.3.1 Request Version.....	66

163	4.1.3.2 Response Version.....	66
164	4.1.3.3 Permissible Version Combinations.....	67
165	4.2 SAML Namespace Version.....	67
166	4.2.1 Schema Evolution.....	67
167	5 SAML and XML Signature Syntax and Processing.....	68
168	5.1 Signing Assertions.....	68
169	5.2 Request/Response Signing.....	68
170	5.3 Signature Inheritance.....	68
171	5.4 XML Signature Profile.....	69
172	5.4.1 Signing Formats and Algorithms.....	69
173	5.4.2 References.....	69
174	5.4.3 Canonicalization Method.....	69
175	5.4.4 Transforms.....	69
176	5.4.5 KeyInfo.....	70
177	5.4.6 Example.....	70
178	6 SAML and XML Encryption Syntax and Processing.....	73
179	6.1 General Considerations.....	73
180	6.2 Combining Signatures and Encryption.....	73
181	7 SAML Extensibility.....	74
182	7.1 Schema Extension.....	74
183	7.1.1 Assertion Schema Extension.....	74
184	7.1.2 Protocol Schema Extension.....	74
185	7.2 Schema Wildcard Extension Points.....	75
186	7.2.1 Assertion Extension Points.....	75
187	7.2.2 Protocol Extension Points.....	75
188	7.3 Identifier Extension.....	75
189	8 SAML-Defined Identifiers.....	76
190	8.1 Action Namespace Identifiers.....	76
191	8.1.1 Read/Write/Execute/Delete/Control.....	76
192	8.1.2 Read/Write/Execute/Delete/Control with Negation.....	76
193	8.1.3 Get/Head/Put/Post.....	77
194	8.1.4 UNIX File Permissions.....	77
195	8.2 Attribute Name Format Identifiers.....	77
196	8.2.1 Unspecified.....	77
197	8.2.2 URI Reference.....	78
198	8.2.3 Basic.....	78
199	8.3 Name Identifier Format Identifiers.....	78
200	8.3.1 Unspecified.....	78
201	8.3.2 Email Address.....	78
202	8.3.3 X.509 Subject Name.....	78
203	8.3.4 Windows Domain Qualified Name.....	78
204	8.3.5 Kerberos Principal Name.....	79
205	8.3.6 Entity Identifier.....	79
206	8.3.7 Persistent Identifier.....	79
207	8.3.8 Transient Identifier.....	80
208	8.4 Consent Identifiers.....	80
209	8.4.1 Unspecified.....	80
210	8.4.2 Obtained.....	80
211	8.4.3 Prior.....	80
212	8.4.4 Implicit.....	81
213	8.4.5 Explicit.....	81

214	8.4.6 Unavailable.....	81
215	8.4.7 Inapplicable.....	81
216	9 References.....	82
217	9.1 Normative References.....	82
218	9.2 Non-Normative References.....	82
219	Appendix A. Acknowledgments.....	84
220	Appendix B. Notices.....	86
221		

# 1 Introduction

The Security Assertion Markup Language (SAML) defines the syntax and processing semantics of assertions made about a subject by a system entity. In the course of making, or relying upon such assertions, SAML system entities may use other protocols to communicate either regarding an assertion itself, or the subject of an assertion. This specification defines both the structure of SAML assertions, and an associated set of protocols, in addition to the processing rules involved in managing a SAML system.

SAML assertions and protocol messages are encoded in XML [XML] and use XML namespaces [XMLNS]. They are typically embedded in other structures for transport, such as HTTP POST requests or XML-encoded SOAP messages. The SAML bindings specification [SAMLBind] provides frameworks for the embedding and transport of SAML protocol messages. The SAML profiles specification [SAMLProf] provides a baseline set of profiles for the use of SAML assertions and protocols to accomplish specific use cases or achieve interoperability when using SAML features.

For additional explanation of SAML terms and concepts, refer to the SAML technical overview [SAMLTechOvw] and the SAML glossary [SAMLGloss]. Files containing just the SAML assertion schema [SAML-XSD] and protocol schema [SAMPLP-XSD] are also available. The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

The following sections describe how to understand the rest of this specification.

## 1.1 Notation

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC 2119].

Listings of SAML schemas appear like this.

Example code listings appear like this.

**Note:** Notes like this are sometimes used to highlight non-normative commentary.

This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. In cases of disagreement between the SAML schema documents and schema listings in this specification, the schema documents take precedence. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace, defined in a schema [SAML-XSD]. The prefix is generally elided in mentions of SAML assertion-related elements in text.
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace, defined in a schema [SAMPLP-XSD]. The prefix is generally elided in mentions of XML protocol-related elements in text.
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema [XMLSig-XSD].

Prefix	XML Namespace	Comments
xenc:	http://www.w3.org/2001/04/xmlenc#	This namespace is defined in the XML Encryption Syntax and Processing specification [XMLEnc] and its governing schema [XMLEnc-XSD].
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This namespace is defined in the W3C XML Schema specification [Schema1] for schema-related markup that appears in XML instances.

255 This specification uses the following typographical conventions in text: <SAML*Element*>,  
 256 <ns:ForeignElement>, XMLAttribute, **Datatype**, OtherKeyword.

## 257 1.2 Schema Organization and Namespaces

258 The SAML assertion structures are defined in a schema [SAML-XSD] associated with the following XML  
 259 namespace:

260 urn:oasis:names:tc:SAML:2.0:assertion

261 The SAML request-response protocol structures are defined in a schema [SAML-P-XSD] associated with  
 262 the following XML namespace:

263 urn:oasis:names:tc:SAML:2.0:protocol

264 The assertion schema is imported into the protocol schema. See Section 4.2 for information on SAML  
 265 namespace versioning.

266 Also imported into both schemas is the schema for XML Signature [XMLSig], which is associated with the  
 267 following XML namespace:

268 http://www.w3.org/2000/09/xmldsig#

269 Finally, the schema for XML Encryption [XMLEnc] is imported into the assertion schema and is associated  
 270 with the following XML namespace:

271 http://www.w3.org/2001/04/xmlenc#

## 272 1.3 Common Data Types

273 The following sections define how to use and interpret common data types that appear throughout the  
 274 SAML schemas.

### 275 1.3.1 String Values

276 All SAML string values have the type **xs:string**, which is built in to the W3C XML Schema Datatypes  
 277 specification [Schema2]. Unless otherwise noted in this specification or particular profiles, all strings in  
 278 SAML messages MUST consist of at least one non-whitespace character (whitespace is defined in the  
 279 XML Recommendation [XML] Section 2.3).

280 Unless otherwise noted in this specification or particular profiles, all elements in SAML documents that  
 281 have the XML Schema **xs:string** type, or a type derived from that, MUST be compared using an exact  
 282 binary comparison. In particular, SAML implementations and deployments MUST NOT depend on case-  
 283 insensitive string comparisons, normalization or trimming of whitespace, or conversion of locale-specific



formats such as numbers or currency. This requirement is intended to conform to the W3C working-draft Requirements for String Identity, Matching, and String Indexing [W3C-CHAR].

If an implementation is comparing values that are represented using different character encodings, the implementation **MUST** use a comparison method that returns the same result as converting both values to the Unicode character encoding, Normalization Form C [UNICODE-C], and then performing an exact binary comparison. This requirement is intended to conform to the W3C Character Model for the World Wide Web [W3C-CharMod], and in particular the rules for Unicode-normalized Text.

Applications that compare data received in SAML documents to data from external sources **MUST** take into account the normalization rules specified for XML. Text contained within elements is normalized so that line endings are represented using linefeed characters (ASCII code 10<sub>Decimal</sub>), as described in the XML Recommendation [XML] Section 2.11. XML attribute values defined as strings (or types derived from strings) are normalized as described in [XML] Section 3.3.3. All whitespace characters are replaced with blanks (ASCII code 32<sub>Decimal</sub>).

The SAML specification does not define collation or sorting order for XML attribute values or element content. SAML implementations **MUST NOT** depend on specific sorting orders for values, because these can differ depending on the locale settings of the hosts involved.

### 1.3.2 URI Values

All SAML URI reference values have the type **xs:anyURI**, which is built in to the W3C XML Schema Datatypes specification [Schema2].

Unless otherwise indicated in this specification, all URI reference values used within SAML-defined elements or attributes **MUST** consist of at least one non-whitespace character, and are **REQUIRED** to be absolute [RFC 2396].

Note that the SAML specification makes extensive use of URI references as identifiers, such as status codes, format types, attribute and system entity names, etc. In such cases, it is essential that the values be both unique and consistent, such that the same URI is never used at different times to represent different underlying information.

### 1.3.3 Time Values

All SAML time values have the type **xs:dateTime**, which is built in to the W3C XML Schema Datatypes specification [Schema2], and **MUST** be expressed in UTC form, with no time zone component.

SAML system entities **SHOULD NOT** rely on time resolution finer than milliseconds. Implementations **MUST NOT** generate time instants that specify leap seconds.

### 1.3.4 ID and ID Reference Values

The **xs:ID** simple type is used to declare SAML identifiers for assertions, requests, and responses. Values declared to be of type **xs:ID** in this specification **MUST** satisfy the following properties in addition to those imposed by the definition of the **xs:ID** type itself:

- Any party that assigns an identifier **MUST** ensure that there is negligible probability that that party or any other party will accidentally assign the same identifier to a different data object.
- Where a data object declares that it has a particular identifier, there **MUST** be exactly one such declaration.

The mechanism by which a SAML system entity ensures that the identifier is unique is left to the implementation. In the case that a random or pseudorandom technique is employed, the probability of two randomly chosen identifiers being identical **MUST** be less than or equal to  $2^{-128}$  and **SHOULD** be less than or equal to  $2^{-160}$ . This requirement **MAY** be met by encoding a randomly chosen value between 128 and

327 160 bits in length. The encoding must conform to the rules defining the **xs:ID** datatype. A pseudorandom  
328 generator **MUST** be seeded with unique material in order to ensure the desired uniqueness properties  
329 between different systems.

330 The **xs:NCName** simple type is used in SAML to reference identifiers of type **xs:ID** since **xs:IDREF**  
331 cannot be used for this purpose. In SAML, the element referred to by a SAML identifier reference might  
332 actually be defined in a document separate from that in which the identifier reference is used. Using  
333 **xs:IDREF** would violate the requirement that its value match the value of an ID attribute on some element  
334 in the same XML document.

335 **Note:** It is anticipated that the World Wide Web Consortium will standardize a global  
336 attribute for holding ID-typed values, called `xml:id` [XML-ID]. The Security Services  
337 Technical Committee plans to move away from SAML-specific ID attributes to this style of  
338 assigning unique identifiers as soon as practicable after the `xml:id` attribute is  
339 standardized.

## 2 SAML Assertions

An assertion is a package of information that supplies zero or more statements made by a **SAML authority**; SAML authorities are sometimes referred to as **asserting parties** in discussions of assertion generation and exchange, and system entities that use received assertions are known as **relying parties**. (Note that these terms are different from **requester** and **responder**, which are reserved for discussions of SAML protocol message exchange.)

SAML assertions are usually made about a **subject**, represented by the <Subject> element. However, the <Subject> element is optional, and other specifications and profiles may utilize the SAML assertion structure to make similar statements without specifying a subject, or possibly specifying the subject in an alternate way. Typically there are a number of **service providers** that can make use of assertions about a subject in order to control access and provide customized service, and accordingly they become the relying parties of an asserting party called an **identity provider**.

This SAML specification defines three different kinds of assertion statements that can be created by a SAML authority. All SAML-defined statements are associated with a subject. The three kinds of statement defined in this specification are:

- **Authentication:** The assertion subject was authenticated by a particular means at a particular time.
- **Attribute:** The assertion subject is associated with the supplied attributes.
- **Authorization Decision:** A request to allow the assertion subject to access the specified resource has been granted or denied.

The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, attribute, authorization decision, or user-defined statements containing the specifics.

As described in Section 7, extensions are permitted by the SAML assertion schema, allowing user-defined extensions to assertions and statements, as well as allowing the definition of new kinds of assertions and statements.

The SAML technical overview [SAMLTechOvw] and glossary [SAMLGloss] provide more detailed explanation of SAML terms and concepts.

### 2.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the assertion schema:

```
<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  <import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
20020212/xmldsig-core-schema.xsd"/>
  <import namespace="http://www.w3.org/2001/04/xmlenc#"
    schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-
20021210/xenc-schema.xsd"/>
  <annotation>
    <documentation>
      Document identifier: saml-schema-assertion-2.0
```

```

388      Location: http://docs.oasis-open.org/security/saml/v2.0/
389      Revision history:
390      V1.0 (November, 2002):
391      Initial Standard Schema.
392      V1.1 (September, 2003):
393      Updates within the same V1.0 namespace.
394      V2.0 (March, 2005):
395      New assertion schema for SAML V2.0 namespace.
396    </documentation>
397  </annotation>
398  ...
399 </schema>

```

## 2.2 Name Identifiers

The following sections define the SAML constructs that contain descriptive identifiers for subjects and the issuers of assertions and protocol messages.

There are a number of circumstances in SAML in which it is useful for two system entities to communicate regarding a third party; for example, the SAML authentication request protocol enables third-party authentication of a subject. Thus, it is useful to establish a means by which parties may be associated with identifiers that are meaningful to each of the parties. In some cases, it will be necessary to limit the scope within which an identifier is used to a small set of system entities (to preserve the privacy of a subject, for example). Similar identifiers may also be used to refer to the issuer of a SAML protocol message or assertion.

It is possible that two or more system entities may use the same name identifier value when referring to different identities. Thus, each entity may have a different understanding of that same name. SAML provides **name qualifiers** to disambiguate a name identifier by effectively placing it in a federated **namespace** related to the name qualifiers. SAML V2.0 allows an identifier to be qualified in terms of both an asserting party and a particular relying party or affiliation, allowing identifiers to exhibit pair-wise semantics, when required.

Name identifiers may also be encrypted to further improve their privacy-preserving characteristics, particularly in cases where the identifier may be transmitted via an intermediary.

**Note:** To avoid use of relatively advanced XML schema constructs (among other reasons), the various types of identifier elements do not share a common type hierarchy.

### 2.2.1 Element <BaseID>

The <BaseID> element is an extension point that allows applications to add new kinds of identifiers. Its **BaseIDAbstractType** complex type is abstract and is thus usable only as the base of a derived type. It includes the following attributes for use by extended identifier representations:

**NameQualifier** [Optional]

The security or administrative domain that qualifies the identifier. This attribute provides a means to federate identifiers from disparate user stores without collision.

**SPNameQualifier** [Optional]

Further qualifies an identifier with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate identifiers on the basis of the relying party or parties.

The **NameQualifier** and **SPNameQualifier** attributes SHOULD be omitted unless the identifier's type definition explicitly defines their use and semantics.

The following schema fragment defines the `<BaseID>` element and its **BaseIDAbstractType** complex type:

```
<attributeGroup name="IDNameQualifiers">
  <attribute name="NameQualifier" type="string" use="optional"/>
  <attribute name="SPNameQualifier" type="string" use="optional"/>
</attributeGroup>
<element name="BaseID" type="saml:BaseIDAbstractType"/>
<complexType name="BaseIDAbstractType" abstract="true">
  <attributeGroup ref="saml:IDNameQualifiers"/>
</complexType>
```

## 2.2.2 Complex Type NameIDType

The **NameIDType** complex type is used when an element serves to represent an entity by a string-valued name. It is a more restricted form of identifier than the `<BaseID>` element and is the type underlying both the `<NameID>` and `<Issuer>` elements. In addition to the string content containing the actual identifier, it provides the following optional attributes:

### NameQualifier [Optional]

The security or administrative domain that qualifies the name. This attribute provides a means to federate names from disparate user stores without collision.

### SPNameQualifier [Optional]

Further qualifies a name with the name of a service provider or affiliation of providers. This attribute provides an additional means to federate names on the basis of the relying party or parties.

### Format [Optional]

A URI reference representing the classification of string-based identifier information. See Section 8.3 for the SAML-defined URI references that MAY be used as the value of the `Format` attribute and their associated descriptions and processing rules. Unless otherwise specified by an element based on this type, if no `Format` value is provided, then the value `urn:oasis:names:tc:SAML:1.0:nameid-format:unspecified` (see Section 8.3.1) is in effect.

When a `Format` value other than one specified in Section 8.3 is used, the content of an element of this type is to be interpreted according to the definition of that format as provided outside of this specification. If not otherwise indicated by the definition of the format, issues of anonymity, pseudonymity, and the persistence of the identifier with respect to the asserting and relying parties are implementation-specific.

### SPProvidedID [Optional]

A name identifier established by a service provider or affiliation of providers for the entity, if different from the primary name identifier given in the content of the element. This attribute provides a means of integrating the use of SAML with existing identifiers already in use by a service provider. For example, an existing identifier can be "attached" to the entity using the Name Identifier Management protocol defined in Section 3.6.

Additional rules for the content of (or the omission of) these attributes can be defined by elements that make use of this type, and by specific `Format` definitions. The `NameQualifier` and `SPNameQualifier` attributes SHOULD be omitted unless the element or format explicitly defines their use and semantics.

The following schema fragment defines the **NameIDType** complex type:

```
<complexType name="NameIDType">
  <simpleContent>
```

```

479     <extension base="string">
480       <attributeGroup ref="saml:IDNameQualifiers"/>
481       <attribute name="Format" type="anyURI" use="optional"/>
482       <attribute name="SPProvidedID" type="string" use="optional"/>
483     </extension>
484   </simpleContent>
485 </complexType>

```

### 2.2.3 Element <NameID>

The <NameID> element is of type **NameIDType** (see Section 2.2.2), and is used in various SAML assertion constructs such as the <Subject> and <SubjectConfirmation> elements, and in various protocol messages (see Section 3).

The following schema fragment defines the <NameID> element:

```

491 <element name="NameID" type="saml:NameIDType"/>

```

### 2.2.4 Element <EncryptedID>

The <EncryptedID> element is of type **EncryptedElementType**, and carries the content of an unencrypted identifier element in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedID> element contains the following elements:

<xenc:EncryptedData> [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The *Type* attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of **NameIDType** or **AssertionType**, or a type that is derived from **BaseIDAbstractType**, **NameIDType**, or **AssertionType**.

<xenc:EncryptedKey> [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a *Recipient* attribute that specifies the entity for whom the key has been encrypted. The value of the *Recipient* attribute SHOULD be the URI identifier of a SAML system entity, as defined by Section 8.3.6.

Encrypted identifiers are intended as a privacy protection mechanism when the plain-text value passes through an intermediary. As such, the ciphertext MUST be unique to any given encryption operation. For more on such issues, see [XMLEnc] Section 6.3.

Note that an entire assertion can be encrypted into this element and used as an identifier. In such a case, the <Subject> element of the encrypted assertion supplies the "identifier" of the subject of the enclosing assertion. Note also that if the identifying assertion is invalid, then so is the enclosing assertion.

The following schema fragment defines the <EncryptedID> element and its **EncryptedElementType** complex type:

```

515 <complexType name="EncryptedElementType">
516   <sequence>
517     <element ref="xenc:EncryptedData"/>
518     <element ref="xenc:EncryptedKey" minOccurs="0" maxOccurs="unbounded"/>
519   </sequence>
520 </complexType>
521 <element name="EncryptedID" type="saml:EncryptedElementType"/>

```



## 2.2.5 Element <Issuer>

The <Issuer> element, with complex type **NameIDType**, provides information about the issuer of a SAML assertion or protocol message. The element requires the use of a string to carry the issuer's name, but permits various pieces of descriptive data (see Section 2.2.2).

Overriding the usual rule for this element's type, if no `Format` value is provided with this element, then the value `urn:oasis:names:tc:SAML:2.0:nameid-format:entity` is in effect (see Section 8.3.6).

The following schema fragment defines the <Issuer> element:

```
<element name="Issuer" type="saml:NameIDType"/>
```

## 2.3 Assertions

The following sections define the SAML constructs that either contain assertion information or provide a means to refer to an existing assertion.

### 2.3.1 Element <AssertionIDRef>

The <AssertionIDRef> element makes a reference to a SAML assertion by its unique identifier. The specific authority who issued the assertion or from whom the assertion can be obtained is not specified as part of the reference. See Section 3.3.1 for a protocol element that uses such a reference to ask for the corresponding assertion.

The following schema fragment defines the <AssertionIDRef> element:

```
<element name="AssertionIDRef" type="NCName"/>
```

### 2.3.2 Element <AssertionURIRef>

The <AssertionURIRef> element makes a reference to a SAML assertion by URI reference. The URI reference MAY be used to retrieve the corresponding assertion in a manner specific to the URI reference. See Section 3.7 of the Bindings specification [SAMLBind] for information on how this element is used in a protocol binding to accomplish this.

The following schema fragment defines the <AssertionURIRef> element:

```
<element name="AssertionURIRef" type="anyURI"/>
```

### 2.3.3 Element <Assertion>

The <Assertion> element is of the **AssertionType** complex type. This type specifies the basic information that is common to all assertions, including the following elements and attributes:

**Version** [Required]

The version of this assertion. The identifier for the version of SAML defined in this specification is "2.0". SAML versioning is discussed in Section 4.

**ID** [Required]

The identifier for this assertion. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness.

**IssueInstant** [Required]

The time instant of issue in UTC, as described in Section 1.3.3.

558 <Issuer> [Required]  
 559 The SAML authority that is making the claim(s) in the assertion. The issuer SHOULD be unambiguous  
 560 to the intended relying parties.

561 This specification defines no particular relationship between the entity represented by this element  
 562 and the signer of the assertion (if any). Any such requirements imposed by a relying party that  
 563 consumes the assertion or by specific profiles are application-specific.

564 <ds:Signature> [Optional]  
 565 An XML Signature that protects the integrity of and authenticates the issuer of the assertion, as  
 566 described below and in Section 5.

567 <Subject> [Optional]  
 568 The subject of the statement(s) in the assertion.

569 <Conditions> [Optional]  
 570 Conditions that MUST be evaluated when assessing the validity of and/or when using the assertion.  
 571 See Section 2.5 for additional information on how to evaluate conditions.

572 <Advice> [Optional]  
 573 Additional information related to the assertion that assists processing in certain situations but which  
 574 MAY be ignored by applications that do not understand the advice or do not wish to make use of it.

575 Zero or more of the following statement elements:

576 <Statement>  
 577 A statement of a type defined in an extension schema. An `xsi:type` attribute MUST be used to  
 578 indicate the actual statement type.

579 <AuthnStatement>  
 580 An authentication statement.

581 <AuthzDecisionStatement>  
 582 An authorization decision statement.

583 <AttributeStatement>  
 584 An attribute statement.

585 An assertion with no statements MUST contain a <Subject> element. Such an assertion identifies a  
 586 principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further  
 587 information associated with that principal.

588 Otherwise <Subject>, if present, identifies the subject of all of the statements in the assertion. If  
 589 <Subject> is omitted, then the statements in the assertion apply to a subject or subjects identified in an  
 590 application- or profile-specific manner. SAML itself defines no such statements, and an assertion without a  
 591 subject has no defined meaning in this specification.

592 Depending on the requirements of particular protocols or profiles, the issuer of a SAML assertion may  
 593 often need to be authenticated, and integrity protection may often be required. Authentication and  
 594 message integrity MAY be provided by mechanisms provided by a protocol binding in use during the  
 595 delivery of an assertion (see [SAMLBind]). The SAML assertion MAY be signed, which provides both  
 596 authentication of the issuer and integrity protection.

597 If such a signature is used, then the <ds:Signature> element MUST be present, and a relying party  
 598 MUST verify that the signature is valid (that is, that the assertion has not been tampered with) in  
 599 accordance with [XMLSig]. If it is invalid, then the relying party MUST NOT rely on the contents of the  
 600 assertion. If it is valid, then the relying party SHOULD evaluate the signature to determine the identity and  
 601 appropriateness of the issuer and may continue to process the assertion in accordance with this

specification and as it deems appropriate (for example, evaluating conditions, advice, following profile-specific rules, and so on).

Note that whether signed or unsigned, the inclusion of multiple statements within a single assertion is semantically equivalent to a set of assertions containing those statements individually (provided the subject, conditions, etc. are also the same).

The following schema fragment defines the `<Assertion>` element and its **AssertionType** complex type:

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
  <sequence>
    <element ref="saml:Issuer"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="saml:Subject" minOccurs="0"/>
    <element ref="saml:Conditions" minOccurs="0"/>
    <element ref="saml:Advice" minOccurs="0"/>
    <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="saml:Statement"/>
      <element ref="saml:AuthnStatement"/>
      <element ref="saml:AuthzDecisionStatement"/>
      <element ref="saml:AttributeStatement"/>
    </choice>
  </sequence>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

### 2.3.4 Element `<EncryptedAssertion>`

The `<EncryptedAssertion>` element represents an assertion in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAssertion>` element contains the following elements:

`<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of or derived from **AssertionType**.

`<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of the `Recipient` attribute SHOULD be the URI identifier of a SAML system entity as defined by Section 8.3.6.

Encrypted assertions are intended as a confidentiality protection mechanism when the plain-text value passes through an intermediary.

The following schema fragment defines the `<EncryptedAssertion>` element:

```
<element name="EncryptedAssertion" type="saml:EncryptedElementType"/>
```

## 2.4 Subjects

This section defines the SAML constructs used to describe the subject of an assertion.

## 2.4.1 Element <Subject>

The optional <Subject> element specifies the principal that is the subject of all of the (zero or more) statements in the assertion. It contains an identifier, a series of one or more subject confirmations, or both:

<BaseID>, <NameID>, or <EncryptedID> [Optional]

Identifies the subject.

<SubjectConfirmation> [Zero or More]

Information that allows the subject to be confirmed. If more than one subject confirmation is provided, then satisfying any one of them is sufficient to confirm the subject for the purpose of applying the assertion.

A <Subject> element can contain both an identifier and zero or more subject confirmations which a relying party can verify when processing an assertion. If any one of the included subject confirmations are verified, the relying party MAY treat the entity presenting the assertion as one that the asserting party has associated with the principal identified in the name identifier and associated with the statements in the assertion. This attesting entity and the actual subject may or may not be the same entity.

If there are no subject confirmations included, then any relationship between the presenter of the assertion and the actual subject is unspecified.

A <Subject> element SHOULD NOT identify more than one principal.

The following schema fragment defines the <Subject> element and its **SubjectType** complex type:

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
  <choice>
    <sequence>
      <choice>
        <element ref="saml:BaseID"/>
        <element ref="saml:NameID"/>
        <element ref="saml:EncryptedID"/>
      </choice>
      <element ref="saml:SubjectConfirmation" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
  </choice>
</complexType>
```

### 2.4.1.1 Element <SubjectConfirmation>

The <SubjectConfirmation> element provides the means for a relying party to verify the correspondence of the subject of the assertion with the party with whom the relying party is communicating. It contains the following attributes and elements:

Method [Required]

A URI reference that identifies a protocol or mechanism to be used to confirm the subject. URI references identifying SAML-defined confirmation methods are currently defined in the SAML profiles specification [SAMLProf]. Additional methods MAY be added by defining new URIs and profiles or by private agreement.

<BaseID>, <NameID>, or <EncryptedID> [Optional]

Identifies the entity expected to satisfy the enclosing subject confirmation requirements.

692 <SubjectConfirmationData> [Optional]

693 Additional confirmation information to be used by a specific confirmation method. For example, typical  
 694 content of this element might be a <ds:KeyInfo> element as defined in the XML Signature Syntax  
 695 and Processing specification [XMLSig], which identifies a cryptographic key (See also Section  
 696 2.4.1.3). Particular confirmation methods MAY define a schema type to describe the elements,  
 697 attributes, or content that may appear in the <SubjectConfirmationData> element.

698 The following schema fragment defines the <SubjectConfirmation> element and its  
 699 **SubjectConfirmationType** complex type:

```

700 <element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
701 <complexType name="SubjectConfirmationType">
702   <sequence>
703     <choice minOccurs="0">
704       <element ref="saml:BaseID"/>
705       <element ref="saml:NameID"/>
706       <element ref="saml:EncryptedID"/>
707     </choice>
708     <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
709   </sequence>
710   <attribute name="Method" type="anyURI" use="required"/>
711 </complexType>

```

#### 712 2.4.1.2 Element <SubjectConfirmationData>

713 The <SubjectConfirmationData> element has the **SubjectConfirmationDataType** complex type. It  
 714 specifies additional data that allows the subject to be confirmed or constrains the circumstances under  
 715 which the act of subject confirmation can take place. Subject confirmation takes place when a relying  
 716 party seeks to verify the relationship between an entity presenting the assertion (that is, the attesting  
 717 entity) and the subject of the assertion's claims. It contains the following optional attributes that can apply  
 718 to any method:

719 NotBefore [Optional]

720 A time instant before which the subject cannot be confirmed. The time value is encoded in UTC, as  
 721 described in Section 1.3.3.

722 NotOnOrAfter [Optional]

723 A time instant at which the subject can no longer be confirmed. The time value is encoded in UTC, as  
 724 described in Section 1.3.3.

725 Recipient [Optional]

726 A URI specifying the entity or location to which an attesting entity can present the assertion. For  
 727 example, this attribute might indicate that the assertion must be delivered to a particular network  
 728 endpoint in order to prevent an intermediary from redirecting it someplace else.

729 InResponseTo [Optional]

730 The ID of a SAML protocol message in response to which an attesting entity can present the  
 731 assertion. For example, this attribute might be used to correlate the assertion to a SAML request that  
 732 resulted in its presentation.

733 Address [Optional]

734 The network address/location from which an attesting entity can present the assertion. For example,  
 735 this attribute might be used to bind the assertion to particular client addresses to prevent an attacker  
 736 from easily stealing and presenting the assertion from another location. IPv4 addresses SHOULD be  
 737 represented in the usual dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses SHOULD be  
 738 represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g.,  
 739 "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

#### 740 Arbitrary attributes

741 This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary namespace-  
 742 qualified XML attributes to be added to `<SubjectConfirmationData>` constructs without the need  
 743 for an explicit schema extension. This allows additional fields to be added as needed to supply  
 744 additional confirmation-related information. SAML extensions MUST NOT add local (non-namespace-  
 745 qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the  
 746 **SubjectConfirmationDataType** complex type or a derivation of it; such attributes are reserved for  
 747 future maintenance and enhancement of SAML itself.

#### 748 Arbitrary elements

749 This complex type uses an `<xs:any>` extension point to allow arbitrary XML elements to be added to  
 750 `<SubjectConfirmationData>` constructs without the need for an explicit schema extension. This  
 751 allows additional elements to be added as needed to supply additional confirmation-related  
 752 information.

753 Particular confirmation methods and profiles that make use of those methods MAY require the use of one  
 754 or more of the attributes defined within this complex type. For examples of how these attributes (and  
 755 subject confirmation in general) can be used, see the Profiles specification [SAMLProf].

756 Note that the time period specified by the optional `NotBefore` and `NotOnOrAfter` attributes, if present,  
 757 SHOULD fall within the overall assertion validity period as specified by the `<Conditions>` element's  
 758 `NotBefore` and `NotOnOrAfter` attributes. If both attributes are present, the value for `NotBefore`  
 759 MUST be less than (earlier than) the value for `NotOnOrAfter`.

760 The following schema fragment defines the `<SubjectConfirmationData>` element and its  
 761 **SubjectConfirmationDataType** complex type:

```

762 <element name="SubjectConfirmationData"
763   type="saml:SubjectConfirmationDataType"/>
764 <complexType name="SubjectConfirmationDataType" mixed="true">
765   <complexContent>
766     <restriction base="anyType">
767       <sequence>
768         <any namespace="##any" processContents="lax" minOccurs="0"
769 maxOccurs="unbounded"/>
770       </sequence>
771       <attribute name="NotBefore" type="dateTime" use="optional"/>
772       <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
773       <attribute name="Recipient" type="anyURI" use="optional"/>
774       <attribute name="InResponseTo" type="NCName" use="optional"/>
775       <attribute name="Address" type="string" use="optional"/>
776       <anyAttribute namespace="##other" processContents="lax"/>
777     </restriction>
778   </complexContent>
779 </complexType>

```

#### 780 2.4.1.3 Complex Type KeyInfoConfirmationDataType

781 The **KeyInfoConfirmationDataType** complex type constrains a `<SubjectConfirmationData>`  
 782 element to contain one or more `<ds:KeyInfo>` elements that identify cryptographic keys that are used in  
 783 some way to authenticate an attesting entity. The particular confirmation method MUST define the exact  
 784 mechanism by which the confirmation data can be used. The optional attributes defined by the  
 785 **SubjectConfirmationDataType** complex type MAY also appear.

786 This complex type, or a type derived from it, SHOULD be used by any confirmation method that defines its  
 787 confirmation data in terms of the `<ds:KeyInfo>` element.



Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element MUST identify a single cryptographic key. Multiple keys MAY be identified with separate `<ds:KeyInfo>` elements, such as when a principal uses different keys to confirm itself to different relying parties.

The following schema fragment defines the **KeyInfoConfirmationDataType** complex type:

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">
  <complexContent>
    <restriction base="saml:SubjectConfirmationDataType">
      <sequence>
        <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
```

#### 2.4.1.4 Example of a Key-Confirmed <Subject>

To illustrate the way in which the various elements and types fit together, below is an example of a `<Subject>` element containing a name identifier and a subject confirmation based on proof of possession of a key. Note the use of the **KeyInfoConfirmationDataType** to identify the confirmation data syntax as being a `<ds:KeyInfo>` element:

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
    scott@example.org
  </NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
    <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
      <ds:KeyInfo>
        <ds:KeyName>Scott's Key</ds:KeyName>
      </ds:KeyInfo>
    </SubjectConfirmationData>
  </SubjectConfirmation>
</Subject>
```

## 2.5 Conditions

This section defines the SAML constructs that place constraints on the acceptable use of SAML assertions.

### 2.5.1 Element <Conditions>

The `<Conditions>` element MAY contain the following elements and attributes:

**NotBefore** [Optional]

Specifies the earliest time instant at which the assertion is valid. The time value is encoded in UTC, as described in Section 1.3.3.

**NotOnOrAfter** [Optional]

Specifies the time instant at which the assertion has expired. The time value is encoded in UTC, as described in Section 1.3.3.

**<Condition>** [Any Number]

A condition of a type defined in an extension schema. An `xsi:type` attribute MUST be used to indicate the actual condition type.

**<AudienceRestriction>** [Any Number]

Specifies that the assertion is addressed to a particular audience.

834 <OneTimeUse> [Optional]

835 Specifies that the assertion SHOULD be used immediately and MUST NOT be retained for future  
836 use. Although the schema permits multiple occurrences, there MUST be at most one instance of  
837 this element.

838 <ProxyRestriction> [Optional]

839 Specifies limitations that the asserting party imposes on relying parties that wish to subsequently act  
840 as asserting parties themselves and issue assertions of their own on the basis of the information  
841 contained in the original assertion. Although the schema permits multiple occurrences, there MUST  
842 be at most one instance of this element.

843 Because the use of the `xsi:type` attribute would permit an assertion to contain more than one instance  
844 of a SAML-defined subtype of **ConditionsType** (such as **OneTimeUseType**), the schema does not  
845 explicitly limit the number of times particular conditions may be included. A particular type of condition  
846 MAY define limits on such use, as shown above.

847 The following schema fragment defines the <Conditions> element and its **ConditionsType** complex  
848 type:

```
849 <element name="Conditions" type="saml:ConditionsType"/>
850 <complexType name="ConditionsType">
851   <choice minOccurs="0" maxOccurs="unbounded">
852     <element ref="saml:Condition"/>
853     <element ref="saml:AudienceRestriction"/>
854     <element ref="saml:OneTimeUse"/>
855     <element ref="saml:ProxyRestriction"/>
856   </choice>
857   <attribute name="NotBefore" type="dateTime" use="optional"/>
858   <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
859 </complexType>
```

### 860 2.5.1.1 General Processing Rules

861 If an assertion contains a <Conditions> element, then the validity of the assertion is dependent on the  
862 sub-elements and attributes provided, using the following rules in the order shown below.

863 Note that an assertion that has condition validity status **Valid** may nonetheless be untrustworthy or invalid  
864 for reasons such as not being well-formed or schema-valid, not being issued by a trustworthy SAML  
865 authority, or not being authenticated by a trustworthy means.

866 Also note that some conditions may not directly impact the validity of the containing assertion (they always  
867 evaluate to **Valid**), but may restrict the behavior of relying parties with respect to the use of the assertion.

- 868 1. If no sub-elements or attributes are supplied in the <Conditions> element, then the assertion is  
869 considered to be **Valid** with respect to condition processing.
- 870 2. If any sub-element or attribute of the <Conditions> element is determined to be invalid, then the  
871 assertion is considered to be **Invalid**.
- 872 3. If any sub-element or attribute of the <Conditions> element cannot be evaluated, or if an element is  
873 encountered that is not understood, then the validity of the assertion cannot be determined and is  
874 considered to be **Indeterminate**.
- 875 4. If all sub-elements and attributes of the <Conditions> element are determined to be **Valid**, then the  
876 assertion is considered to be **Valid** with respect to condition processing.

877 The first rule that applies terminates condition processing; thus a determination that an assertion is  
878 **Invalid** takes precedence over that of **Indeterminate**.

An assertion that is determined to be **Invalid** or **Indeterminate** MUST be rejected by a relying party (within whatever context or profile it was being processed), just as if the assertion were malformed or otherwise unusable.

### 2.5.1.2 Attributes **NotBefore** and **NotOnOrAfter**

The **NotBefore** and **NotOnOrAfter** attributes specify time limits on the validity of the assertion within the context of its profile(s) of use. They do not guarantee that the statements in the assertion will be correct or accurate throughout the validity period.

The **NotBefore** attribute specifies the time instant at which the validity interval begins. The **NotOnOrAfter** attribute specifies the time instant at which the validity interval has ended.

If the value for either **NotBefore** or **NotOnOrAfter** is omitted, then it is considered unspecified. If the **NotBefore** attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), then the assertion is **Valid** with respect to conditions at any time before the time instant specified by the **NotOnOrAfter** attribute. If the **NotOnOrAfter** attribute is unspecified (and if all other conditions that are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions from the time instant specified by the **NotBefore** attribute with no expiry. If neither attribute is specified (and if any other conditions that are supplied evaluate to **Valid**), the assertion is **Valid** with respect to conditions at any time.

If both attributes are present, the value for **NotBefore** MUST be less than (earlier than) the value for **NotOnOrAfter**.

### 2.5.1.3 Element **<Condition>**

The **<Condition>** element serves as an extension point for new conditions. Its **ConditionAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the **<Condition>** element and its **ConditionAbstractType** complex type:

```
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
```

### 2.5.1.4 Elements **<AudienceRestriction>** and **<Audience>**

The **<AudienceRestriction>** element specifies that the assertion is addressed to one or more specific audiences identified by **<Audience>** elements. Although a SAML relying party that is outside the audiences specified is capable of drawing conclusions from an assertion, the SAML asserting party explicitly makes no representation as to accuracy or trustworthiness to such a party. It contains the following element:

**<Audience>**

A URI reference that identifies an intended audience. The URI reference MAY identify a document that describes the terms and conditions of audience membership. It MAY also contain the unique identifier URI from a SAML name identifier that describes a system entity (see Section 8.3.6).

The audience restriction condition evaluates to **Valid** if and only if the SAML relying party is a member of one or more of the audiences specified.

The SAML asserting party cannot prevent a party to whom the assertion is disclosed from taking action on the basis of the information provided. However, the **<AudienceRestriction>** element allows the SAML asserting party to state explicitly that no warranty is provided to such a party in a machine- and human-readable form. While there can be no guarantee that a court would uphold such a warranty exclusion in every circumstance, the probability of upholding the warranty exclusion is considerably improved.

Note that multiple `<AudienceRestriction>` elements MAY be included in a single assertion, and each MUST be evaluated independently. The effect of this requirement and the preceding definition is that within a given condition, the audiences form a disjunction (an "OR") while multiple conditions form a conjunction (an "AND").

The following schema fragment defines the `<AudienceRestriction>` element and its **AudienceRestrictionType** complex type:

```
<element name="AudienceRestriction"
  type="saml:AudienceRestrictionType"/>
<complexType name="AudienceRestrictionType">
  <complexContent>
    <extension base="saml:ConditionAbstractType">
      <sequence>
        <element ref="saml:Audience" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

### 2.5.1.5 Element `<OneTimeUse>`

In general, relying parties may choose to retain assertions, or the information they contain in some other form, for reuse. The `<OneTimeUse>` condition element allows an authority to indicate that the information in the assertion is likely to change very soon and fresh information should be obtained for each use. An example would be an assertion containing an `<AuthzDecisionStatement>` which was the result of a policy which specified access control which was a function of the time of day.

If system clocks in a distributed environment could be precisely synchronized, then this requirement could be met by careful use of the validity interval. However, since some clock skew between systems will always be present and will be combined with possible transmission delays, there is no convenient way for the issuer to appropriately limit the lifetime of an assertion without running a substantial risk that it will already have expired before it arrives.

The `<OneTimeUse>` element indicates that the assertion SHOULD be used immediately by the relying party and MUST NOT be retained for future use. Relying parties are always free to request a fresh assertion for every use. However, implementations that choose to retain assertions for future use MUST observe the `<OneTimeUse>` element. This condition is independent from the `NotBefore` and `NotOnOrAfter` condition information.

To support the single use constraint, a relying party should maintain a cache of the assertions it has processed containing such a condition. Whenever an assertion with this condition is processed, the cache should be checked to ensure that the same assertion has not been previously received and processed by the relying party.

A SAML authority MUST NOT include more than one `<OneTimeUse>` element within a `<Conditions>` element of an assertion.

For the purposes of determining the validity of the `<Conditions>` element, the `<OneTimeUse>` is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the `<OneTimeUse>` element and its **OneTimeUseType** complex type:

```
<element name="OneTimeUse" type="saml:OneTimeUseType"/>
<complexType name="OneTimeUseType">
  <complexContent>
    <extension base="saml:ConditionAbstractType"/>
  </complexContent>
</complexType>
```

### 2.5.1.6 Element <ProxyRestriction>

Specifies limitations that the asserting party imposes on relying parties that in turn wish to act as asserting parties and issue subsequent assertions of their own on the basis of the information contained in the original assertion. A relying party acting as an asserting party MUST NOT issue an assertion that itself violates the restrictions specified in this condition on the basis of an assertion containing such a condition.

The <ProxyRestriction> element contains the following elements and attributes:

**Count** [Optional]

Specifies the maximum number of indirections that the asserting party permits to exist between this assertion and an assertion which has ultimately been issued on the basis of it.

**<Audience>** [Zero or More]

Specifies the set of audiences to whom the asserting party permits new assertions to be issued on the basis of this assertion.

A **Count** value of zero indicates that a relying party MUST NOT issue an assertion to another relying party on the basis of this assertion. If greater than zero, any assertions so issued MUST themselves contain a <ProxyRestriction> element with a **Count** value of at most one less than this value.

If no <Audience> elements are specified, then no audience restrictions are imposed on the relying parties to whom subsequent assertions can be issued. Otherwise, any assertions so issued MUST themselves contain an <AudienceRestriction> element with at least one of the <Audience> elements present in the previous <ProxyRestriction> element, and no <Audience> elements present that were not in the previous <ProxyRestriction> element.

A SAML authority MUST NOT include more than one <ProxyRestriction> element within a <Conditions> element of an assertion.

For the purposes of determining the validity of the <Conditions> element, the <ProxyRestriction> condition is considered to always be valid. That is, this condition does not affect validity but is a condition on use.

The following schema fragment defines the <ProxyRestriction> element and its **ProxyRestrictionType** complex type:

```
<element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
<complexType name="ProxyRestrictionType">
  <complexContent>
    <extension base="saml:ConditionAbstractType">
      <sequence>
        <element ref="saml:Audience" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="Count" type="nonNegativeInteger" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

## 2.6 Advice

This section defines the SAML constructs that contain additional information about an assertion that an asserting party wishes to provide to a relying party.

## 2.6.1 Element <Advice>

The <Advice> element contains any additional information that the SAML authority wishes to provide. This information MAY be ignored by applications without affecting either the semantics or the validity of the assertion.

The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>, <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in other non-SAML namespaces.

Following are some potential uses of the <Advice> element:

- Include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions).
- State a proof of the assertion claims.
- Specify the timing and distribution points for updates to the assertion.

The following schema fragment defines the <Advice> element and its **AdviceType** complex type:

```
<element name="Advice" type="saml:AdviceType"/>
<complexType name="AdviceType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="saml:AssertionIDRef"/>
    <element ref="saml:AssertionURIRef"/>
    <element ref="saml:Assertion"/>
    <element ref="saml:EncryptedAssertion"/>
    <any namespace="##other" processContents="lax"/>
  </choice>
</complexType>
```

## 2.7 Statements

The following sections define the SAML constructs that contain statement information.

### 2.7.1 Element <Statement>

The <Statement> element is an extension point that allows other assertion-based applications to reuse the SAML assertion framework. SAML itself derives its core statements from this extension point. Its **StatementAbstractType** complex type is abstract and is thus usable only as the base of a derived type.

The following schema fragment defines the <Statement> element and its **StatementAbstractType** complex type:

```
<element name="Statement" type="saml:StatementAbstractType"/>
<complexType name="StatementAbstractType" abstract="true"/>
```

### 2.7.2 Element <AuthnStatement>

The <AuthnStatement> element describes a statement by the SAML authority asserting that the assertion subject was authenticated by a particular means at a particular time. Assertions containing <AuthnStatement> elements MUST contain a <Subject> element.

It is of type **AuthnStatementType**, which extends **StatementAbstractType** with the addition of the following elements and attributes:

**Note:** The <AuthorityBinding> element and its corresponding type were removed from <AuthnStatement> for V2.0 of SAML.



1055 `AuthnInstant` [Required]

1056 Specifies the time at which the authentication took place. The time value is encoded in UTC, as  
 1057 described in Section 1.3.3.

1058 `SessionIndex` [Optional]

1059 Specifies the index of a particular session between the principal identified by the subject and the  
 1060 authenticating authority.

1061 `SessionNotOnOrAfter` [Optional]

1062 Specifies a time instant at which the session between the principal identified by the subject and the  
 1063 SAML authority issuing this statement MUST be considered ended. The time value is encoded in  
 1064 UTC, as described in Section 1.3.3. There is no required relationship between this attribute and a  
 1065 `NotOnOrAfter` condition attribute that may be present in the assertion.

1066 `<SubjectLocality>` [Optional]

1067 Specifies the DNS domain name and IP address for the system from which the assertion subject was  
 1068 apparently authenticated.

1069 `<AuthnContext>` [Required]

1070 The context used by the authenticating authority up to and including the authentication event that  
 1071 yielded this statement. Contains an authentication context class reference, an authentication context  
 1072 declaration or declaration reference, or both. See the Authentication Context specification  
 1073 [SAMLAuthnCxt] for a full description of authentication context information.

1074 In general, any string value MAY be used as a `SessionIndex` value. However, when privacy is a  
 1075 consideration, care must be taken to ensure that the `SessionIndex` value does not invalidate other  
 1076 privacy mechanisms. Accordingly, the value SHOULD NOT be usable to correlate activity by a principal  
 1077 across different session participants. Two solutions that achieve this goal are provided below and are  
 1078 RECOMMENDED:

- 1079 • Use small positive integers (or reoccurring constants in a list) for the `SessionIndex`. The SAML  
 1080 authority SHOULD choose the range of values such that the cardinality of any one integer will be  
 1081 sufficiently high to prevent a particular principal's actions from being correlated across multiple session  
 1082 participants. The SAML authority SHOULD choose values for `SessionIndex` randomly from within  
 1083 this range (except when required to ensure unique values for subsequent statements given to the  
 1084 same session participant but as part of a distinct session).
- 1085 • Use the enclosing assertion's `ID` value in the `SessionIndex`.

1086 The following schema fragment defines the `<AuthnStatement>` element and its **AuthnStatementType**  
 1087 complex type:

```

1088 <element name="AuthnStatement" type="saml:AuthnStatementType"/>
1089 <complexType name="AuthnStatementType">
1090   <complexContent>
1091     <extension base="saml:StatementAbstractType">
1092       <sequence>
1093         <element ref="saml:SubjectLocality" minOccurs="0"/>
1094         <element ref="saml:AuthnContext"/>
1095       </sequence>
1096       <attribute name="AuthnInstant" type="dateTime" use="required"/>
1097       <attribute name="SessionIndex" type="string" use="optional"/>
1098       <attribute name="SessionNotOnOrAfter" type="dateTime"
1099 use="optional"/>
1100     </extension>
1101   </complexContent>
1102 </complexType>

```

### 2.7.2.1 Element <SubjectLocality>

The <SubjectLocality> element specifies the DNS domain name and IP address for the system from which the assertion subject was authenticated. It has the following attributes:

Address [Optional]

The network address of the system from which the principal identified by the subject was authenticated. IPv4 addresses SHOULD be represented in dotted-decimal format (e.g., "1.2.3.4"). IPv6 addresses SHOULD be represented as defined by Section 2.2 of IETF RFC 3513 [RFC 3513] (e.g., "FEDC:BA98:7654:3210:FEDC:BA98:7654:3210").

DNSName [Optional]

The DNS name of the system from which the principal identified by the subject was authenticated.

This element is entirely advisory, since both of these fields are quite easily "spoofed," but may be useful information in some applications.

The following schema fragment defines the <SubjectLocality> element and its **SubjectLocalityType** complex type:

```
<element name="SubjectLocality" type="saml:SubjectLocalityType"/>
<complexType name="SubjectLocalityType">
  <attribute name="Address" type="string" use="optional"/>
  <attribute name="DNSName" type="string" use="optional"/>
</complexType>
```

### 2.7.2.2 Element <AuthnContext>

The <AuthnContext> element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both. Its complex **AuthnContextType** has the following elements:

<AuthnContextClassRef> [Optional]

A URI reference identifying an authentication context class that describes the authentication context declaration that follows.

<AuthnContextDecl> or <AuthnContextDeclRef> [Optional]

Either an authentication context declaration provided by value, or a URI reference that identifies such a declaration. The URI reference MAY directly resolve into an XML document containing the referenced declaration.

<AuthenticatingAuthority> [Zero or More]

Zero or more unique identifiers of authentication authorities that were involved in the authentication of the principal (not including the assertion issuer, who is presumed to have been involved without being explicitly named here).

See the Authentication Context specification [SAMLAuthnCxt] for a full description of authentication context information.

The following schema fragment defines the <AuthnContext> element and its **AuthnContextType** complex type:

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
  <sequence>
    <choice>
      <sequence>
        <element ref="saml:AuthnContextClassRef"/>
        <choice minOccurs="0">
```

```

1148         <element ref="saml:AuthnContextDecl"/>
1149         <element ref="saml:AuthnContextDeclRef"/>
1150     </choice>
1151 </sequence>
1152 <choice>
1153     <element ref="saml:AuthnContextDecl"/>
1154     <element ref="saml:AuthnContextDeclRef"/>
1155 </choice>
1156 </choice>
1157 <element ref="saml:AuthenticatingAuthority" minOccurs="0"
1158 maxOccurs="unbounded"/>
1159 </sequence>
1160 </complexType>
1161 <element name="AuthnContextClassRef" type="anyURI"/>
1162 <element name="AuthnContextDeclRef" type="anyURI"/>
1163 <element name="AuthnContextDecl" type="anyType"/>
1164 <element name="AuthenticatingAuthority" type="anyURI"/>

```

### 2.7.3 Element <AttributeStatement>

The <AttributeStatement> element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

It is of type **AttributeStatementType**, which extends **StatementAbstractType** with the addition of the following elements:

<Attribute> or <EncryptedAttribute> [One or More]

The <Attribute> element specifies an attribute of the assertion subject. An encrypted SAML attribute may be included with the <EncryptedAttribute> element.

The following schema fragment defines the <AttributeStatement> element and its **AttributeStatementType** complex type:

```

1176 <element name="AttributeStatement" type="saml:AttributeStatementType"/>
1177 <complexType name="AttributeStatementType">
1178     <complexContent>
1179         <extension base="saml:StatementAbstractType">
1180             <choice maxOccurs="unbounded">
1181                 <element ref="saml:Attribute"/>
1182                 <element ref="saml:EncryptedAttribute"/>
1183             </choice>
1184         </extension>
1185     </complexContent>
1186 </complexType>

```

#### 2.7.3.1 Element <Attribute>

The <Attribute> element identifies an attribute by name and optionally includes its value(s). It has the **AttributeType** complex type. It is used within an attribute statement to express particular attributes and values associated with an assertion subject, as described in the previous section. It is also used in an attribute query to request that the values of specific SAML attributes be returned (see Section 3.3.2.3 for more information). The <Attribute> element contains the following XML attributes:

Name [Required]

The name of the attribute.

NameFormat [Optional]

A URI reference representing the classification of the attribute name for purposes of interpreting the

name. See Section 8.2 for some URI references that MAY be used as the value of the `NameFormat` attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified` (see Section 8.2.1) is in effect.

#### FriendlyName [Optional]

A string that provides a more human-readable form of the attribute's name, which may be useful in cases in which the actual `Name` is complex or opaque, such as an OID or a UUID. This attribute's value MUST NOT be used as a basis for formally identifying SAML attributes.

#### Arbitrary attributes

This complex type uses an `<xs:anyAttribute>` extension point to allow arbitrary XML attributes to be added to `<Attribute>` constructs without the need for an explicit schema extension. This allows additional fields to be added as needed to supply additional parameters to be used, for example, in an attribute query. SAML extensions MUST NOT add local (non-namespace-qualified) XML attributes or XML attributes qualified by a SAML-defined namespace to the **AttributeType** complex type or a derivation of it; such attributes are reserved for future maintenance and enhancement of SAML itself.

#### `<AttributeValue>` [Any Number]

Contains a value of the attribute. If an attribute contains more than one discrete value, it is RECOMMENDED that each value appear in its own `<AttributeValue>` element. If more than one `<AttributeValue>` element is supplied for an attribute, and any of the elements have a datatype assigned through `xsi:type`, then all of the `<AttributeValue>` elements must have the identical datatype assigned.

The meaning of an `<Attribute>` element that contains no `<AttributeValue>` elements depends on its context. Within an `<AttributeStatement>`, if the SAML attribute exists but has no values, then the `<AttributeValue>` element MUST be omitted. Within a `<samlp:AttributeQuery>`, the absence of values indicates that the requester is interested in any or all of the named attribute's values (see also Section 3.3.2.3).

Any other uses of the `<Attribute>` element by profiles or other specifications MUST define the semantics of specifying or omitting `<AttributeValue>` elements.

The following schema fragment defines the `<Attribute>` element and its **AttributeType** complex type:

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
  <sequence>
    <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Name" type="string" use="required"/>
  <attribute name="NameFormat" type="anyURI" use="optional"/>
  <attribute name="FriendlyName" type="string" use="optional"/>
  <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

#### 2.7.3.1.1 Element `<AttributeValue>`

The `<AttributeValue>` element supplies the value of a specified SAML attribute. It is of the **xs:anyType** type, which allows any well-formed XML to appear as the content of the element.

If the data content of an `<AttributeValue>` element is of an XML Schema simple type (such as **xs:integer** or **xs:string**), the datatype MAY be declared explicitly by means of an `xsi:type` declaration in the `<AttributeValue>` element. If the attribute value contains structured data, the necessary data elements MAY be defined in an extension schema.

**Note:** Specifying a datatype other than an XML Schema simple type on `<AttributeValue>` using `xsi:type` will require the presence of the extension schema that defines the datatype in order for schema processing to proceed.

If a SAML attribute includes an empty value, such as the empty string, the corresponding `<AttributeValue>` element MUST be empty (generally this is serialized as `<AttributeValue/>`). This overrides the requirement in Section 1.3.1 that string values in SAML content contain at least one non-whitespace character.

If a SAML attribute includes a "null" value, the corresponding `<AttributeValue>` element MUST be empty and MUST contain the reserved `xsi:nil` XML attribute with a value of "true" or "1".

The following schema fragment defines the `<AttributeValue>` element:

```
<element name="AttributeValue" type="anyType" nillable="true"/>
```

### 2.7.3.2 Element `<EncryptedAttribute>`

The `<EncryptedAttribute>` element represents a SAML attribute in encrypted fashion, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `<EncryptedAttribute>` element contains the following elements:

`<xenc:EncryptedData>` [Required]

The encrypted content and associated encryption details, as defined by the XML Encryption Syntax and Processing specification [XMLEnc]. The `Type` attribute SHOULD be present and, if present, MUST contain a value of <http://www.w3.org/2001/04/xmlenc#Element>. The encrypted content MUST contain an element that has a type of or derived from **AttributeType**.

`<xenc:EncryptedKey>` [Zero or More]

Wrapped decryption keys, as defined by [XMLEnc]. Each wrapped key SHOULD include a `Recipient` attribute that specifies the entity for whom the key has been encrypted. The value of the `Recipient` attribute SHOULD be the URI identifier of a system entity with a SAML name identifier, as defined by Section 8.3.6.

Encrypted attributes are intended as a confidentiality protection when the plain-text value passes through an intermediary.

The following schema fragment defines the `<EncryptedAttribute>` element:

```
<element name="EncryptedAttribute" type="saml:EncryptedElementType"/>
```

### 2.7.4 Element `<AuthzDecisionStatement>`

**Note:** The `<AuthzDecisionStatement>` feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

The `<AuthzDecisionStatement>` element describes a statement by the SAML authority asserting that a request for access by the assertion subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence. Assertions containing `<AuthzDecisionStatement>` elements MUST contain a `<Subject>` element.

The resource is identified by means of a URI reference. In order for the assertion to be interpreted correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different

1284 authorization decisions depending on the encoding of the resource URI reference. Rules for normalizing  
 1285 URI references are to be found in IETF RFC 2396 [RFC 2396] Section 6:

1286 In general, the rules for equivalence and definition of a normal form, if any, are scheme  
 1287 dependent. When a scheme uses elements of the common syntax, it will also use the common  
 1288 syntax equivalence rules, namely that the scheme and hostname are case insensitive and a URL  
 1289 with an explicit ":port", where the port is the default for the scheme, is equivalent to one where  
 1290 the port is elided.

1291 To avoid ambiguity resulting from variations in URI encoding, SAML system entities SHOULD employ the  
 1292 URI normalized form wherever possible as follows:

- 1293 • SAML authorities SHOULD encode all resource URI references in normalized form.
- 1294 • Relying parties SHOULD convert resource URI references to normalized form prior to processing.

1295 Inconsistent URI reference interpretation can also result from differences between the URI reference  
 1296 syntax and the semantics of an underlying file system. Particular care is required if URI references are  
 1297 employed to specify an access control policy language. The following security conditions SHOULD be  
 1298 satisfied by the system which employs SAML assertions:

- 1299 • Parts of the URI reference syntax are case sensitive. If the underlying file system is case insensitive,  
 1300 a requester SHOULD NOT be able to gain access to a denied resource by changing the case of a  
 1301 part of the resource URI reference.
- 1302 • Many file systems support mechanisms such as logical paths and symbolic links, which allow users  
 1303 to establish logical equivalences between file system entries. A requester SHOULD NOT be able to  
 1304 gain access to a denied resource by creating such an equivalence.

1305 The `<AuthzDecisionStatement>` element is of type **AuthzDecisionStatementType**, which extends  
 1306 **StatementAbstractType** with the addition of the following elements and attributes:

1307 **Resource** [Required]

1308 A URI reference identifying the resource to which access authorization is sought. This attribute MAY  
 1309 have the value of the empty URI reference (""), and the meaning is defined to be "the start of the  
 1310 current document", as specified by IETF RFC 2396 [RFC 2396] Section 4.2.

1311 **Decision** [Required]

1312 The decision rendered by the SAML authority with respect to the specified resource. The value is of  
 1313 the **DecisionType** simple type.

1314 **<Action>** [One or more]

1315 The set of actions authorized to be performed on the specified resource.

1316 **<Evidence>** [Optional]

1317 A set of assertions that the SAML authority relied on in making the decision.

1318 The following schema fragment defines the `<AuthzDecisionStatement>` element and its  
 1319 **AuthzDecisionStatementType** complex type:

```

1320 <element name="AuthzDecisionStatement"
1321   type="saml:AuthzDecisionStatementType"/>
1322 <complexType name="AuthzDecisionStatementType">
1323   <complexContent>
1324     <extension base="saml:StatementAbstractType">
1325       <sequence>
1326         <element ref="saml:Action" maxOccurs="unbounded"/>
1327         <element ref="saml:Evidence" minOccurs="0"/>
1328       </sequence>
1329       <attribute name="Resource" type="anyURI" use="required"/>
1330       <attribute name="Decision" type="saml:DecisionType" use="required"/>
  
```



```

1331     </extension>
1332   </complexContent>
1333 </complexType>

```

#### 2.7.4.1 Simple Type DecisionType

The **DecisionType** simple type defines the possible values to be reported as the status of an authorization decision statement.

Permit

The specified action is permitted.

Deny

The specified action is denied.

Indeterminate

The SAML authority cannot determine whether the specified action is permitted or denied.

The **Indeterminate** decision value is used in situations where the SAML authority requires the ability to provide an affirmative statement but where it is not able to issue a decision. Additional information as to the reason for the refusal or inability to provide a decision MAY be returned as **<StatusDetail>** elements in the enclosing **<Response>**.

The following schema fragment defines the **DecisionType** simple type:

```

1348 <simpleType name="DecisionType">
1349   <restriction base="string">
1350     <enumeration value="Permit"/>
1351     <enumeration value="Deny"/>
1352     <enumeration value="Indeterminate"/>
1353   </restriction>
1354 </simpleType>

```

#### 2.7.4.2 Element <Action>

The **<Action>** element specifies an action on the specified resource for which permission is sought. Its string-data content provides the label for an action sought to be performed on the specified resource, and it has the following attribute:

Namespace [Optional]

A URI reference representing the namespace in which the name of the specified action is to be interpreted. If this element is absent, the namespace

**urn:oasis:names:tc:SAML:1.0:action:rwedc-negation** specified in Section 8.1.2 is in effect.

The following schema fragment defines the **<Action>** element and its **ActionType** complex type:

```

1365 <element name="Action" type="saml:ActionType"/>
1366 <complexType name="ActionType">
1367   <simpleContent>
1368     <extension base="string">
1369       <attribute name="Namespace" type="anyURI" use="required"/>
1370     </extension>
1371   </simpleContent>
1372 </complexType>

```

### 2.7.4.3 Element <Evidence>

The <Evidence> element contains one or more assertions or assertion references that the SAML authority relied on in issuing the authorization decision. It has the **EvidenceType** complex type. It contains a mixture of one or more of the following elements:

<AssertionIDRef> [Any number]

Specifies an assertion by reference to the value of the assertion's ID attribute.

<AssertionURIRef> [Any number]

Specifies an assertion by means of a URI reference.

<Assertion> [Any number]

Specifies an assertion by value.

<EncryptedAssertion> [Any number]

Specifies an encrypted assertion by value.

Providing an assertion as evidence MAY affect the reliance agreement between the SAML relying party and the SAML authority making the authorization decision. For example, in the case that the SAML relying party presented an assertion to the SAML authority in a request, the SAML authority MAY use that assertion as evidence in making its authorization decision without endorsing the <Evidence> element's assertion as valid either to the relying party or any other third party.

The following schema fragment defines the <Evidence> element and its **EvidenceType** complex type:

```
<element name="Evidence" type="saml:EvidenceType"/>
<complexType name="EvidenceType">
  <choice maxOccurs="unbounded">
    <element ref="saml:AssertionIDRef"/>
    <element ref="saml:AssertionURIRef"/>
    <element ref="saml:Assertion"/>
    <element ref="saml:EncryptedAssertion"/>
  </choice>
</complexType>
```

### 3 SAML Protocols

SAML protocol messages can be generated and exchanged using a variety of protocols. The SAML bindings specification [SAMLBind] describes specific means of transporting protocol messages using existing widely deployed transport protocols. The SAML profile specification [SAMLProf] describes a number of applications of the protocols defined in this section together with additional processing rules, restrictions, and requirements that facilitate interoperability.

Specific SAML request and response messages derive from common types. The requester sends an element derived from **RequestAbstractType** to a SAML responder, and the responder generates an element adhering to or deriving from **StatusResponseType**, as shown in Figure 1.

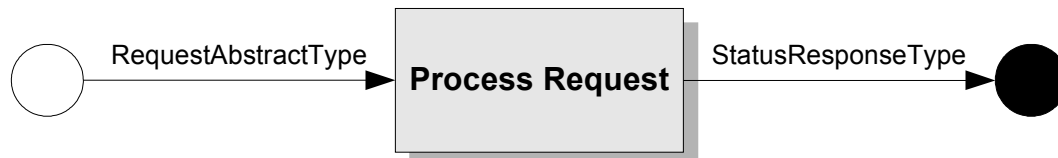


Figure 1: SAML Request-Response Protocol

In certain cases, when permitted by profiles, a SAML response MAY be generated and sent without the responder having received a corresponding request.

The protocols defined by SAML achieve the following actions:

- Returning one or more requested assertions. This can occur in response to either a direct request for specific assertions or a query for assertions that meet particular criteria.
- Performing authentication on request and returning the corresponding assertion
- Registering a name identifier or terminating a name registration on request
- Retrieving a protocol message that has been requested by means of an artifact
- Performing a near-simultaneous logout of a collection of related sessions (“single logout”) on request
- Providing a name identifier mapping on request

Throughout this section, text descriptions of elements and types in the SAML protocol namespace are not shown with the conventional namespace prefix `samlp:.` For clarity, text descriptions of elements and types in the SAML assertion namespace are indicated with the conventional namespace prefix `saml:.`

#### 3.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the protocol schema:

```

<schema
  targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified"
  blockDefault="substitution"
  version="2.0">
  
```

```

1439 <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
1440   schemaLocation="saml-schema-assertion-2.0.xsd"/>
1441 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1442   schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
1443 20020212/xmldsig-core-schema.xsd"/>
1444 <annotation>
1445   <documentation>
1446     Document identifier: saml-schema-protocol-2.0
1447     Location: http://docs.oasis-open.org/security/saml/v2.0/
1448     Revision history:
1449     V1.0 (November, 2002):
1450       Initial Standard Schema.
1451     V1.1 (September, 2003):
1452       Updates within the same V1.0 namespace.
1453     V2.0 (March, 2005):
1454       New protocol schema based in a SAML V2.0 namespace.
1455   </documentation>
1456 </annotation>
1457 ...
1458 </schema>

```

## 1459 3.2 Requests and Responses

1460 The following sections define the SAML constructs and basic requirements that underlie all of the request  
 1461 and response messages used in SAML protocols.

### 1462 3.2.1 Complex Type RequestAbstractType

1463 All SAML requests are of types that are derived from the abstract **RequestAbstractType** complex type.  
 1464 This type defines common attributes and elements that are associated with all SAML requests:

1465 **Note:** The <RespondWith> element has been removed from **RequestAbstractType**  
 1466 for V2.0 of SAML.

#### 1467 ID [Required]

1468 An identifier for the request. It is of type **xs:ID** and MUST follow the requirements specified in Section  
 1469 1.3.4 for identifier uniqueness. The values of the **ID** attribute in a request and the **InResponseTo**  
 1470 attribute in the corresponding response MUST match.

#### 1471 Version [Required]

1472 The version of this request. The identifier for the version of SAML defined in this specification is "2.0".  
 1473 SAML versioning is discussed in Section 4.

#### 1474 IssueInstant [Required]

1475 The time instant of issue of the request. The time value is encoded in UTC, as described in Section  
 1476 1.3.3.

#### 1477 Destination [Optional]

1478 A URI reference indicating the address to which this request has been sent. This is useful to prevent  
 1479 malicious forwarding of requests to unintended recipients, a protection that is required by some  
 1480 protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the  
 1481 location at which the message was received. If it does not, the request MUST be discarded. Some  
 1482 protocol bindings may require the use of this attribute (see [SAMLBind]).

#### 1483 Consent [Optional]

1484 Indicates whether or not (and under what conditions) consent has been obtained from a principal in  
 1485 the sending of this request. See Section 8.4 for some URI references that MAY be used as the value

1486 of the `Consent` attribute and their associated descriptions. If no `Consent` value is provided, the  
 1487 identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in  
 1488 effect.

1489 `<saml:Issuer>` [Optional]

1490 Identifies the entity that generated the request message. (For more information on this element, see  
 1491 Section 2.2.5.)

1492 `<ds:Signature>` [Optional]

1493 An XML Signature that authenticates the requester and provides message integrity, as described  
 1494 below and in Section 5.

1495 `<Extensions>` [Optional]

1496 This extension point contains optional protocol message extension elements that are agreed on  
 1497 between the communicating parties. No extension schema is required in order to make use of this  
 1498 extension point, and even if one is provided, the lax validation setting does not impose a requirement  
 1499 for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-  
 1500 SAML-defined namespace.

1501 Depending on the requirements of particular protocols or profiles, a SAML requester may often need to  
 1502 authenticate itself, and message integrity may often be required. Authentication and message integrity  
 1503 MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML request  
 1504 MAY be signed, which provides both authentication of the requester and message integrity.

1505 If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML  
 1506 responder MUST verify that the signature is valid (that is, that the message has not been tampered with)  
 1507 in accordance with [XMLSig]. If it is invalid, then the responder MUST NOT rely on the contents of the  
 1508 request and SHOULD respond with an error. If it is valid, then the responder SHOULD evaluate the  
 1509 signature to determine the identity and appropriateness of the signer and may continue to process the  
 1510 request or respond with an error (if the request is invalid for some other reason).

1511 If a `Consent` attribute is included and the value indicates that some form of principal consent has been  
 1512 obtained, then the request SHOULD be signed.

1513 If a SAML responder deems a request to be invalid according to SAML syntax or processing rules, then if  
 1514 it responds, it MUST return a SAML response message with a `<StatusCode>` element with the value  
 1515 `urn:oasis:names:tc:SAML:2.0:status:Requester`. In some cases, for example during a  
 1516 suspected denial-of-service attack, not responding at all may be warranted.

1517 The following schema fragment defines the **RequestAbstractType** complex type:

```

1518 <complexType name="RequestAbstractType" abstract="true">
1519   <sequence>
1520     <element ref="saml:Issuer" minOccurs="0"/>
1521     <element ref="ds:Signature" minOccurs="0"/>
1522     <element ref="samlp:Extensions" minOccurs="0"/>
1523   </sequence>
1524   <attribute name="ID" type="ID" use="required"/>
1525   <attribute name="Version" type="string" use="required"/>
1526   <attribute name="IssueInstant" type="dateTime" use="required"/>
1527   <attribute name="Destination" type="anyURI" use="optional"/>
1528   <attribute name="Consent" type="anyURI" use="optional"/>
1529 </complexType>
1530 <element name="Extensions" type="samlp:ExtensionsType"/>
1531 <complexType name="ExtensionsType">
1532   <sequence>
1533     <any namespace="##other" processContents="lax" maxOccurs="unbounded"/>
1534   </sequence>
1535 </complexType>

```

### 3.2.2 Complex Type **StatusResponseType**

All SAML responses are of types that are derived from the **StatusResponseType** complex type. This type defines common attributes and elements that are associated with all SAML responses:

#### **ID** [Required]

An identifier for the response. It is of type **xs:ID**, and MUST follow the requirements specified in Section 1.3.4 for identifier uniqueness.

#### **InResponseTo** [Optional]

A reference to the identifier of the request to which the response corresponds, if any. If the response is not generated in response to a request, or if the **ID** attribute value of a request cannot be determined (for example, the request is malformed), then this attribute MUST NOT be present. Otherwise, it MUST be present and its value MUST match the value of the corresponding request's **ID** attribute.

#### **Version** [Required]

The version of this response. The identifier for the version of SAML defined in this specification is "2.0". SAML versioning is discussed in Section 4.

#### **IssueInstant** [Required]

The time instant of issue of the response. The time value is encoded in UTC, as described in Section 1.3.3.

#### **Destination** [Optional]

A URI reference indicating the address to which this response has been sent. This is useful to prevent malicious forwarding of responses to unintended recipients, a protection that is required by some protocol bindings. If it is present, the actual recipient MUST check that the URI reference identifies the location at which the message was received. If it does not, the response MUST be discarded. Some protocol bindings may require the use of this attribute (see [SAMLBind]).

#### **Consent** [Optional]

Indicates whether or not (and under what conditions) consent has been obtained from a principal in the sending of this response. See Section 8.4 for some URI references that MAY be used as the value of the **Consent** attribute and their associated descriptions. If no **Consent** value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:consent:unspecified` (see Section 8.4.1) is in effect.

#### **<saml:Issuer>** [Optional]

Identifies the entity that generated the response message. (For more information on this element, see Section 2.2.5.)

#### **<ds:Signature>** [Optional]

An XML Signature that authenticates the responder and provides message integrity, as described below and in Section 5.

#### **<Extensions>** [Optional]

This extension point contains optional protocol message extension elements that are agreed on between the communicating parties. . No extension schema is required in order to make use of this extension point, and even if one is provided, the lax validation setting does not impose a requirement for the extension to be valid. SAML extension elements MUST be namespace-qualified in a non-SAML-defined namespace.

#### **<Status>** [Required]

A code representing the status of the corresponding request.



Depending on the requirements of particular protocols or profiles, a SAML responder may often need to authenticate itself, and message integrity may often be required. Authentication and message integrity MAY be provided by mechanisms provided by the protocol binding (see [SAMLBind]). The SAML response MAY be signed, which provides both authentication of the responder and message integrity.

If such a signature is used, then the `<ds:Signature>` element MUST be present, and the SAML requester receiving the response MUST verify that the signature is valid (that is, that the message has not been tampered with) in accordance with [XMLSig]. If it is invalid, then the requester MUST NOT rely on the contents of the response and SHOULD treat it as an error. If it is valid, then the requester SHOULD evaluate the signature to determine the identity and appropriateness of the signer and may continue to process the response as it deems appropriate.

If a `Consent` attribute is included and the value indicates that some form of principal consent has been obtained, then the response SHOULD be signed.

The following schema fragment defines the **StatusResponseType** complex type:

```
<complexType name="StatusResponseType">
  <sequence>
    <element ref="saml:Issuer" minOccurs="0"/>
    <element ref="ds:Signature" minOccurs="0"/>
    <element ref="samlp:Extensions" minOccurs="0"/>
    <element ref="samlp:Status"/>
  </sequence>
  <attribute name="ID" type="ID" use="required"/>
  <attribute name="InResponseTo" type="NCName" use="optional"/>
  <attribute name="Version" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
  <attribute name="Destination" type="anyURI" use="optional"/>
  <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

### 3.2.2.1 Element `<Status>`

The `<Status>` element contains the following elements:

`<StatusCode>` [Required]

A code representing the status of the activity carried out in response to the corresponding request.

`<StatusMessage>` [Optional]

A message which MAY be returned to an operator.

`<StatusDetail>` [Optional]

Additional information concerning the status of the request.

The following schema fragment defines the `<Status>` element and its **StatusType** complex type:

```
<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
  <sequence>
    <element ref="samlp:StatusCode"/>
    <element ref="samlp:StatusMessage" minOccurs="0"/>
    <element ref="samlp:StatusDetail" minOccurs="0"/>
  </sequence>
</complexType>
```

### 3.2.2.2 Element `<StatusCode>`

The `<StatusCode>` element specifies a code or a set of nested codes representing the status of the corresponding request. The `<StatusCode>` element has the following element and attribute:

1627 Value [Required]

1628 The status code value. This attribute contains a URI reference. The value of the topmost  
1629 <StatusCode> element MUST be from the top-level list provided in this section.

1630 <StatusCode> [Optional]

1631 A subordinate status code that provides more specific information on an error condition. Note that  
1632 responders MAY omit subordinate status codes in order to prevent attacks that seek to probe for  
1633 additional information by intentionally presenting erroneous requests.

1634 The permissible top-level <StatusCode> values are as follows:

1635 urn:oasis:names:tc:SAML:2.0:status:Success

1636 The request succeeded. Additional information MAY be returned in the <StatusMessage> and/or  
1637 <StatusDetail> elements.

1638 urn:oasis:names:tc:SAML:2.0:status:Requester

1639 The request could not be performed due to an error on the part of the requester.

1640 urn:oasis:names:tc:SAML:2.0:status:Responder

1641 The request could not be performed due to an error on the part of the SAML responder or SAML  
1642 authority.

1643 urn:oasis:names:tc:SAML:2.0:status:VersionMismatch

1644 The SAML responder could not process the request because the version of the request message was  
1645 incorrect.

1646 The following second-level status codes are referenced at various places in this specification. Additional  
1647 second-level status codes MAY be defined in future versions of the SAML specification. System entities  
1648 are free to define more specific status codes by defining appropriate URI references.

1649 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

1650 The responding provider was unable to successfully authenticate the principal.

1651 urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

1652 Unexpected or invalid content was encountered within a <saml:Attribute> or  
1653 <saml:AttributeValue> element.

1654 urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

1655 The responding provider cannot or will not support the requested name identifier policy.

1656 urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

1657 The specified authentication context requirements cannot be met by the responder.

1658 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

1659 Used by an intermediary to indicate that none of the supported identity provider <Loc> elements in an  
1660 <IDPList> can be resolved or that none of the supported identity providers are available.

1661 urn:oasis:names:tc:SAML:2.0:status:NoPassive

1662 Indicates the responding provider cannot authenticate the principal passively, as has been requested.

1663 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

1664 Used by an intermediary to indicate that none of the identity providers in an <IDPList> are  
1665 supported by the intermediary.

1666 urn:oasis:names:tc:SAML:2.0:status:PartialLogout  
 1667       Used by a session authority to indicate to a session participant that it was not able to propagate logout  
 1668       to all other session participants.

1669 urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded  
 1670       Indicates that a responding provider cannot authenticate the principal directly and is not permitted to  
 1671       proxy the request further.

1672 urn:oasis:names:tc:SAML:2.0:status:RequestDenied  
 1673       The SAML responder or SAML authority is able to process the request but has chosen not to respond.  
 1674       This status code MAY be used when there is concern about the security context of the request  
 1675       message or the sequence of request messages received from a particular requester.

1676 urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported  
 1677       The SAML responder or SAML authority does not support the request.

1678 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated  
 1679       The SAML responder cannot process any requests with the protocol version specified in the request.

1680 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh  
 1681       The SAML responder cannot process the request because the protocol version specified in the  
 1682       request message is a major upgrade from the highest protocol version supported by the responder.

1683 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow  
 1684       The SAML responder cannot process the request because the protocol version specified in the  
 1685       request message is too low.

1686 urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized  
 1687       The resource value provided in the request message is invalid or unrecognized.

1688 urn:oasis:names:tc:SAML:2.0:status:TooManyResponses  
 1689       The response message would contain more elements than the SAML responder is able to return.

1690 urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile  
 1691       An entity that has no knowledge of a particular attribute profile has been presented with an attribute  
 1692       drawn from that profile.

1693 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal  
 1694       The responding provider does not recognize the principal specified or implied by the request.

1695 urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding  
 1696       The SAML responder cannot properly fulfill the request using the protocol binding specified in the  
 1697       request.

1698 The following schema fragment defines the <StatusCode> element and its **StatusCodeType** complex  
 1699 type:

```

1700 <element name="StatusCode" type="samlp:StatusCodeType"/>
1701 <complexType name="StatusCodeType">
1702   <sequence>
1703     <element ref="samlp:StatusCode" minOccurs="0"/>
1704   </sequence>
1705   <attribute name="Value" type="anyURI" use="required"/>
1706 </complexType>

```

### 3.2.2.3 Element <StatusMessage>

The <StatusMessage> element specifies a message that MAY be returned to an operator:

The following schema fragment defines the <StatusMessage> element:

```
<element name="StatusMessage" type="string"/>
```

### 3.2.2.4 Element <StatusDetail>

The <StatusDetail> element MAY be used to specify additional information concerning the status of the request. The additional information consists of zero or more elements from any namespace, with no requirement for a schema to be present or for schema validation of the <StatusDetail> contents.

The following schema fragment defines the <StatusDetail> element and its **StatusDetailType** complex type:

```
<element name="StatusDetail" type="samlp:StatusDetailType"/>
<complexType name="StatusDetailType">
  <sequence>
    <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

## 3.3 Assertion Query and Request Protocol

This section defines messages and processing rules for requesting existing assertions by reference or querying for assertions by subject and statement type.

### 3.3.1 Element <AssertionIDRequest>

If the requester knows the unique identifier of one or more assertions, the <AssertionIDRequest> message element can be used to request that they be returned in a <Response> message. The <saml:AssertionIDRef> element is used to specify each assertion to return. See Section 2.3.1 for more information on this element.

The following schema fragment defines the <AssertionIDRequest> element:

```
<element name="AssertionIDRequest" type="samlp:AssertionIDRequestType"/>
<complexType name="AssertionIDRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="saml:AssertionIDRef" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.2 Queries

The following sections define the SAML query request messages.

#### 3.3.2.1 Element <SubjectQuery>

The <SubjectQuery> message element is an extension point that allows new SAML queries to be defined that specify a single SAML subject. Its **SubjectQueryAbstractType** complex type is abstract and

is thus usable only as the base of a derived type. **SubjectQueryAbstractType** adds the `<saml:Subject>` element (defined in Section 2.4) to **RequestAbstractType**.

The following schema fragment defines the `<SubjectQuery>` element and its **SubjectQueryAbstractType** complex type:

```
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="saml:Subject"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.2.2 Element `<AuthnQuery>`

The `<AuthnQuery>` message element is used to make the query "What assertions containing authentication statements are available for this subject?" A successful `<Response>` will contain one or more assertions containing authentication statements.

The `<AuthnQuery>` message MUST NOT be used as a request for a new authentication using credentials provided in the request. `<AuthnQuery>` is a request for statements about authentication acts that have occurred in a previous interaction between the indicated subject and the authentication authority.

This element is of type **AuthnQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element and attribute:

**SessionIndex** [Optional]

If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject within the context of the supplied session information?"

**<RequestedAuthnContext>** [Optional]

If present, specifies a filter for possible responses. Such a query asks the question "What assertions containing authentication statements do you have for this subject that satisfy the authentication context requirements in this element?"

In response to an authentication query, a SAML authority returns assertions with authentication statements as follows:

- Rules given in Section 3.3.4 for matching against the `<Subject>` element of the query identify the assertions that may be returned.
- If the **SessionIndex** attribute is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions MUST contain a **SessionIndex** attribute that matches the **SessionIndex** attribute in the query. It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.
- If the `<RequestedAuthnContext>` element is present in the query, at least one `<AuthnStatement>` element in the set of returned assertions MUST contain an `<AuthnContext>` element that satisfies the element in the query (see Section 3.3.2.2.1). It is OPTIONAL for the complete set of all such matching assertions to be returned in the response.

The following schema fragment defines the `<AuthnQuery>` element and its **AuthnQueryType** complex type:

```
<element name="AuthnQuery" type="samlp:AuthnQueryType"/>
```

```

1794 <complexType name="AuthnQueryType">
1795   <complexContent>
1796     <extension base="samlp:SubjectQueryAbstractType">
1797       <sequence>
1798         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
1799       </sequence>
1800       <attribute name="SessionIndex" type="string" use="optional"/>
1801     </extension>
1802   </complexContent>
1803 </complexType>

```

### 1804 3.3.2.2.1 Element <RequestedAuthnContext>

1805 The <RequestedAuthnContext> element specifies the authentication context requirements of  
 1806 authentication statements returned in response to a request or query. Its **RequestedAuthnContextType**  
 1807 complex type defines the following elements and attributes:

1808 <saml:AuthnContextClassRef> or <saml:AuthnContextDeclRef> [One or More]

1809 Specifies one or more URI references identifying authentication context classes or declarations.  
 1810 These elements are defined in Section 2.7.2.2. For more information about authentication context  
 1811 classes, see [SAMLAuthnCxt].

1812 Comparison [Optional]

1813 Specifies the comparison method used to evaluate the requested context classes or statements, one  
 1814 of "exact", "minimum", "maximum", or "better". The default is "exact".

1815 Either a set of class references or a set of declaration references can be used. The set of supplied  
 1816 references MUST be evaluated as an ordered set, where the first element is the most preferred  
 1817 authentication context class or declaration. If none of the specified classes or declarations can be satisfied  
 1818 in accordance with the rules below, then the responder MUST return a <Response> message with a  
 1819 second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext.

1820 If Comparison is set to "exact" or omitted, then the resulting authentication context in the authentication  
 1821 statement MUST be the exact match of at least one of the authentication contexts specified.

1822 If Comparison is set to "minimum", then the resulting authentication context in the authentication  
 1823 statement MUST be at least as strong (as deemed by the responder) as one of the authentication  
 1824 contexts specified.

1825 If Comparison is set to "better", then the resulting authentication context in the authentication  
 1826 statement MUST be stronger (as deemed by the responder) than any one of the authentication contexts  
 1827 specified.

1828 If Comparison is set to "maximum", then the resulting authentication context in the authentication  
 1829 statement MUST be as strong as possible (as deemed by the responder) without exceeding the strength  
 1830 of at least one of the authentication contexts specified.

1831 The following schema fragment defines the <RequestedAuthnContext> element and its  
 1832 **RequestedAuthnContextType** complex type:

```

1833 <element name="RequestedAuthnContext" type="samlp:RequestedAuthnContextType"/>
1834 <complexType name="RequestedAuthnContextType">
1835   <choice>
1836     <element ref="saml:AuthnContextClassRef" maxOccurs="unbounded"/>
1837     <element ref="saml:AuthnContextDeclRef" maxOccurs="unbounded"/>
1838   </choice>
1839   <attribute name="Comparison" type="samlp:AuthnContextComparisonType"
1840 use="optional"/>
1841 </complexType>
1842 <simpleType name="AuthnContextComparisonType">

```



```

1843     <restriction base="string">
1844         <enumeration value="exact"/>
1845         <enumeration value="minimum"/>
1846         <enumeration value="maximum"/>
1847         <enumeration value="better"/>
1848     </restriction>
1849 </simpleType>

```

### 3.3.2.3 Element <AttributeQuery>

The <AttributeQuery> element is used to make the query "Return the requested attributes for this subject." A successful response will be in the form of assertions containing attribute statements, to the extent allowed by policy. This element is of type **AttributeQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following element:

<saml:Attribute> [Any Number]

Each <saml:Attribute> element specifies an attribute whose value(s) are to be returned. If no attributes are specified, it indicates that all attributes allowed by policy are requested. If a given <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then if that attribute is returned in the response, it MUST NOT contain any values that are not equal to the values specified in the query. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on <saml:Attribute>, see Section 2.7.3.1.

A single query MUST NOT contain two <saml:Attribute> elements with the same Name and NameFormat values (that is, a given attribute MUST be named only once in a query).

In response to an attribute query, a SAML authority returns assertions with attribute statements as follows:

- Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the assertions that may be returned.
- If any <Attribute> elements are present in the query, they constrain/filter the attributes and optionally the values returned, as noted above.
- The attributes and values returned MAY also be constrained by application-specific policy considerations.

The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to indicate problems with the interpretation of attribute or value information in a query.

The following schema fragment defines the <AttributeQuery> element and its **AttributeQueryType** complex type:

```

1877 <element name="AttributeQuery" type="samlp:AttributeQueryType"/>
1878 <complexType name="AttributeQueryType">
1879     <complexContent>
1880         <extension base="samlp:SubjectQueryAbstractType">
1881             <sequence>
1882                 <element ref="saml:Attribute" minOccurs="0"
1883 maxOccurs="unbounded"/>
1884             </sequence>
1885         </extension>
1886     </complexContent>
1887 </complexType>

```

### 3.3.2.4 Element <AuthzDecisionQuery>

The <AuthzDecisionQuery> element is used to make the query “Should these actions on this resource be allowed for this subject, given this evidence?” A successful response will be in the form of assertions containing authorization decision statements.

**Note:** The <AuthzDecisionQuery> feature has been frozen as of SAML V2.0, with no future enhancements planned. Users who require additional functionality may want to consider the eXtensible Access Control Markup Language [XACML], which offers enhanced authorization decision features.

This element is of type **AuthzDecisionQueryType**, which extends **SubjectQueryAbstractType** with the addition of the following elements and attribute:

**Resource** [Required]

A URI reference indicating the resource for which authorization is requested.

**<saml:Action>** [One or More]

The actions for which authorization is requested. For more information on this element, see Section 2.7.4.2.

**<saml:Evidence>** [Optional]

A set of assertions that the SAML authority MAY rely on in making its authorization decision. For more information on this element, see Section 2.7.4.3.

In response to an authorization decision query, a SAML authority returns assertions with authorization decision statements as follows:

- Rules given in Section 3.3.4 for matching against the <Subject> element of the query identify the assertions that may be returned.

The following schema fragment defines the <AuthzDecisionQuery> element and its **AuthzDecisionQueryType** complex type:

```
<element name="AuthzDecisionQuery" type="samlp:AuthzDecisionQueryType"/>
<complexType name="AuthzDecisionQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:Action" maxOccurs="unbounded"/>
        <element ref="saml:Evidence" minOccurs="0"/>
      </sequence>
      <attribute name="Resource" type="anyURI" use="required"/>
    </extension>
  </complexContent>
</complexType>
```

### 3.3.3 Element <Response>

The <Response> message element is used when a response consists of a list of zero or more assertions that satisfy the request. It has the complex type **ResponseType**, which extends **StatusResponseType** and adds the following elements:

**<saml:Assertion>** or **<saml:EncryptedAssertion>** [Any Number]

Specifies an assertion by value, or optionally an encrypted assertion by value. See Section 2.3.3 for more information on these elements.

The following schema fragment defines the <Response> element and its **ResponseType** complex type:

```

1932 <element name="Response" type="samlp:ResponseType"/>
1933 <complexType name="ResponseType">
1934   <complexContent>
1935     <extension base="samlp:StatusResponseType">
1936       <choice minOccurs="0" maxOccurs="unbounded">
1937         <element ref="saml:Assertion"/>
1938         <element ref="saml:EncryptedAssertion"/>
1939       </choice>
1940     </extension>
1941   </complexContent>
1942 </complexType>

```

### 3.3.4 Processing Rules

In response to a SAML-defined query message, every assertion returned by a SAML authority **MUST** contain a `<saml:Subject>` element that **strongly matches** the `<saml:Subject>` element found in the query.

A `<saml:Subject>` element S1 strongly matches S2 if and only if the following two conditions both apply:

- If S2 includes an identifier element (`<BaseID>`, `<NameID>`, or `<EncryptedID>`), then S1 **MUST** include an identical identifier element, but the element **MAY** be encrypted (or not) in either S1 or S2. In other words, the decrypted form of the identifier **MUST** be identical in S1 and S2. "Identical" means that the identifier element's content and attribute values **MUST** be the same. An encrypted identifier will be identical to the original according to this definition, once decrypted.
- If S2 includes one or more `<saml:SubjectConfirmation>` elements, then S1 **MUST** include at least one `<saml:SubjectConfirmation>` element such that S1 can be confirmed in the manner described by at least one `<saml:SubjectConfirmation>` element in S2.

As an example of what is and is not permitted, S1 could contain a `<saml:NameID>` with a particular `Format` value, and S2 could contain a `<saml:EncryptedID>` element that is the result of encrypting S1's `<saml:NameID>` element. However, S1 and S2 cannot contain a `<saml:NameID>` element with different `Format` values and element content, even if the two identifiers are considered to refer to the same principal.

If the SAML authority cannot provide an assertion with any statements satisfying the constraints expressed by a query or assertion reference, the `<Response>` element **MUST NOT** contain an `<Assertion>` element and **MUST** include a `<StatusCode>` element with the value `urn:oasis:names:tc:SAML:2.0:status:Success`.

All other processing rules associated with the underlying request and response messages **MUST** be observed.

## 3.4 Authentication Request Protocol

When a principal (or an agent acting on the principal's behalf) wishes to obtain assertions containing authentication statements to establish a security context at one or more relying parties, it can use the authentication request protocol to send an `<AuthnRequest>` message element to a SAML authority and request that it return a `<Response>` message containing one or more such assertions. Such assertions **MAY** contain additional statements of any type, but at least one assertion **MUST** contain at least one authentication statement. A SAML authority that supports this protocol is also termed an identity provider.

Apart from this requirement, the specific contents of the returned assertions depend on the profile or context of use. Also, the exact means by which the principal or agent authenticates to the identity provider is not specified, though the means of authentication might impact the content of the response. Other issues related to the validation of authentication credentials by the identity provider or any communication

1979 between the identity provider and any other entities involved in the authentication process are also out of  
1980 scope of this protocol.

1981 The descriptions and processing rules in the following sections reference the following actors, many of  
1982 whom might be the same entity in a particular profile of use:

1983 Requester

1984 The entity who creates the authentication request and to whom the response is to be returned.

1985 Presenter

1986 The entity who presents the request to the identity provider and either authenticates itself during  
1987 the transmission of the message, or relies on an existing security context to establish its identity. If  
1988 not the requester, the presenter acts as an intermediary between the requester and the  
1989 responding identity provider.

1990 Requested Subject

1991 The entity about whom one or more assertions are being requested.

1992 Attesting Entity

1993 The entity or entities expected to be able to satisfy one of the `<SubjectConfirmation>`  
1994 elements of the resulting assertion(s).

1995 Relying Party

1996 The entity or entities expected to consume the assertion(s) to accomplish a purpose defined by  
1997 the profile or context of use, generally to establish a security context.

1998 Identity Provider

1999 The entity to whom the presenter gives the request and from whom the presenter receives the  
2000 response.

2001 **3.4.1 Element `<AuthnRequest>`**

2002 To request that an identity provider issue an assertion with an authentication statement, a presenter  
2003 authenticates to that identity provider (or relies on an existing security context) and sends it an  
2004 `<AuthnRequest>` message that describes the properties that the resulting assertion needs to have to  
2005 satisfy its purpose. Among these properties may be information that relates to the content of the assertion  
2006 and/or information that relates to how the resulting `<Response>` message should be delivered to the  
2007 requester. The process of authentication of the presenter may take place before, during, or after the initial  
2008 delivery of the `<AuthnRequest>` message.

2009 The requester might not be the same as the presenter of the request if, for example, the requester is a  
2010 relying party that intends to use the resulting assertion to authenticate or authorize the requested subject  
2011 so that the relying party can decide whether to provide a service.

2012 The `<AuthnRequest>` message SHOULD be signed or otherwise authenticated and integrity protected  
2013 by the protocol binding used to deliver the message.

2014 This message has the complex type **AuthnRequestType**, which extends **RequestAbstractType** and  
2015 adds the following elements and attributes, all of which are optional in general, but may be required by  
2016 specific profiles:

2017 `<saml:Subject>` [Optional]

2018 Specifies the requested subject of the resulting assertion(s). This may include one or more  
2019 `<saml:SubjectConfirmation>` elements to indicate how and/or by whom the resulting assertions  
2020 can be confirmed. For more information on this element, see Section 2.4.

2021 If entirely omitted or if no identifier is included, the presenter of the message is presumed to be the  
 2022 requested subject. If no `<saml:SubjectConfirmation>` elements are included, then the presenter  
 2023 is presumed to be the only attesting entity required and the method is implied by the profile of use  
 2024 and/or the policies of the identity provider.

2025 `<NameIDPolicy>` [Optional]

2026 Specifies constraints on the name identifier to be used to represent the requested subject. If omitted,  
 2027 then any type of identifier supported by the identity provider for the requested subject can be used,  
 2028 constrained by any relevant deployment-specific policies, with respect to privacy, for example.

2029 `<saml:Conditions>` [Optional]

2030 Specifies the SAML conditions the requester expects to limit the validity and/or use of the resulting  
 2031 assertion(s). The responder MAY modify or supplement this set as it deems necessary. The  
 2032 information in this element is used as input to the process of constructing the assertion, rather than as  
 2033 conditions on the use of the request itself. (For more information on this element, see Section 2.5.)

2034 `<RequestedAuthnContext>` [Optional]

2035 Specifies the requirements, if any, that the requester places on the authentication context that applies  
 2036 to the responding provider's authentication of the presenter. See Section 3.3.2.2.1 for processing rules  
 2037 regarding this element.

2038 `<Scoping>` [Optional]

2039 Specifies a set of identity providers trusted by the requester to authenticate the presenter, as well as  
 2040 limitations and context related to proxying of the `<AuthnRequest>` message to subsequent identity  
 2041 providers by the responder.

2042 `ForceAuthn` [Optional]

2043 A Boolean value. If "true", the identity provider MUST authenticate the presenter directly rather than  
 2044 rely on a previous security context. If a value is not provided, the default is "false". However, if both  
 2045 `ForceAuthn` and `IsPassive` are "true", the identity provider MUST NOT freshly authenticate the  
 2046 presenter unless the constraints of `IsPassive` can be met.

2047 `IsPassive` [Optional]

2048 A Boolean value. If "true", the identity provider and the user agent itself MUST NOT visibly take control  
 2049 of the user interface from the requester and interact with the presenter in a noticeable fashion. If a  
 2050 value is not provided, the default is "false".

2051 `AssertionConsumerServiceIndex` [Optional]

2052 Indirectly identifies the location to which the `<Response>` message should be returned to the  
 2053 requester. It applies only to profiles in which the requester is different from the presenter, such as the  
 2054 Web Browser SSO profile in [SAMLProf]. The identity provider MUST have a trusted means to map  
 2055 the index value in the attribute to a location associated with the requester. [SAMLMeta] provides one  
 2056 possible mechanism. If omitted, then the identity provider MUST return the `<Response>` message to  
 2057 the default location associated with the requester for the profile of use. If the index specified is invalid,  
 2058 then the identity provider MAY return an error `<Response>` or it MAY use the default location. This  
 2059 attribute is mutually exclusive with the `AssertionConsumerServiceURL` and `ProtocolBinding`  
 2060 attributes.

2061 `AssertionConsumerServiceURL` [Optional]

2062 Specifies by value the location to which the `<Response>` message MUST be returned to the  
 2063 requester. The responder MUST ensure by some means that the value specified is in fact associated  
 2064 with the requester. [SAMLMeta] provides one possible mechanism; signing the enclosing  
 2065 `<AuthnRequest>` message is another. This attribute is mutually exclusive with the  
 2066 `AssertionConsumerServiceIndex` attribute and is typically accompanied by the  
 2067 `ProtocolBinding` attribute.

## 2068 ProtocolBinding [Optional]

2069 A URI reference that identifies a SAML protocol binding to be used when returning the <Response>  
 2070 message. See [SAMLBind] for more information about protocol bindings and URI references defined  
 2071 for them. This attribute is mutually exclusive with the AssertionConsumerServiceIndex attribute  
 2072 and is typically accompanied by the AssertionConsumerServiceURL attribute.

## 2073 AttributeConsumingServiceIndex [Optional]

2074 Indirectly identifies information associated with the requester describing the SAML attributes the  
 2075 requester desires or requires to be supplied by the identity provider in the <Response> message. The  
 2076 identity provider MUST have a trusted means to map the index value in the attribute to information  
 2077 associated with the requester. [SAMLMeta] provides one possible mechanism. The identity provider  
 2078 MAY use this information to populate one or more <saml:AttributeStatement> elements in the  
 2079 assertion(s) it returns.

## 2080 ProviderName [Optional]

2081 Specifies the human-readable name of the requester for use by the presenter's user agent or the  
 2082 identity provider.

2083 See Section 3.4.1.4 for general processing rules regarding this message.

2084 The following schema fragment defines the <AuthnRequest> element and its **AuthnRequestType**  
 2085 complex type:

```

2086 <element name="AuthnRequest" type="samlp:AuthnRequestType"/>
2087 <complexType name="AuthnRequestType">
2088   <complexContent>
2089     <extension base="samlp:RequestAbstractType">
2090       <sequence>
2091         <element ref="saml:Subject" minOccurs="0"/>
2092         <element ref="samlp:NameIDPolicy" minOccurs="0"/>
2093         <element ref="saml:Conditions" minOccurs="0"/>
2094         <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
2095         <element ref="samlp:Scoping" minOccurs="0"/>
2096       </sequence>
2097       <attribute name="ForceAuthn" type="boolean" use="optional"/>
2098       <attribute name="IsPassive" type="boolean" use="optional"/>
2099       <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
2100       <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
2101 use="optional"/>
2102       <attribute name="AssertionConsumerServiceURL" type="anyURI"
2103 use="optional"/>
2104       <attribute name="AttributeConsumingServiceIndex"
2105 type="unsignedShort" use="optional"/>
2106       <attribute name="ProviderName" type="string" use="optional"/>
2107     </extension>
2108   </complexContent>
2109 </complexType>

```

## 2110 3.4.1.1 Element &lt;NameIDPolicy&gt;

2111 The <NameIDPolicy> element tailors the name identifier in the subjects of assertions resulting from an  
 2112 <AuthnRequest>. Its **NameIDPolicyType** complex type defines the following attributes:

## 2113 Format [Optional]

2114 Specifies the URI reference corresponding to a name identifier format defined in this or another  
 2115 specification (see Section 8.3 for examples). The additional value of  
 2116 urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted is defined specifically for use  
 2117 within this attribute to indicate a request that the resulting identifier be encrypted.



## 2118 SPNameQualifier [Optional]

2119 Optionally specifies that the assertion subject's identifier be returned (or created) in the namespace of  
 2120 a service provider other than the requester, or in the namespace of an affiliation group of service  
 2121 providers. See for example the definition of `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2122 `format:persistent` in Section 8.3.7.

## 2123 AllowCreate [Optional]

2124 A Boolean value used to indicate whether the identity provider is allowed, in the course of fulfilling the  
 2125 request, to create a new identifier to represent the principal. Defaults to "false". When "false", the  
 2126 requester constrains the identity provider to only issue an assertion to it if an acceptable identifier for  
 2127 the principal has already been established. Note that this does not prevent the identity provider from  
 2128 creating such identifiers outside the context of this specific request (for example, in advance for a  
 2129 large number of principals).

2130 When this element is used, if the content is not understood by or acceptable to the identity provider, then a  
 2131 `<Response>` message element MUST be returned with an error `<Status>`, and MAY contain a second-  
 2132 level `<StatusCode>` of `urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy`.

2133 If the `Format` value is omitted or set to `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2134 `format:unspecified`, then the identity provider is free to return any kind of identifier, subject to any  
 2135 additional constraints due to the content of this element or the policies of the identity provider or principal.

2136 The special `Format` value `urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted` indicates  
 2137 that the resulting assertion(s) MUST contain `<EncryptedID>` elements instead of plaintext. The  
 2138 underlying name identifier's unencrypted form can be of any type supported by the identity provider for the  
 2139 requested subject.

2140 Regardless of the `Format` in the `<NameIDPolicy>`, the identity provider MAY return an  
 2141 `<EncryptedID>` in the resulting assertion subject if the policies in effect at the identity provider (possibly  
 2142 specific to the service provider) require that an encrypted identifier be used.

2143 Note that if the requester wishes to permit the identity provider to establish a new identifier for the principal  
 2144 if none exists, it MUST include this element with the `AllowCreate` attribute set to "true". Otherwise,  
 2145 only a principal for whom the identity provider has previously established an identifier usable by the  
 2146 requester can be authenticated successfully. This is primarily useful in conjunction with the  
 2147 `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent` `Format` value (see Section 8.3.7).

2148 The following schema fragment defines the `<NameIDPolicy>` element and its **NameIDPolicyType**  
 2149 complex type:

```
2150 <element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
2151 <complexType name="NameIDPolicyType">
2152   <attribute name="Format" type="anyURI" use="optional"/>
2153   <attribute name="SPNameQualifier" type="string" use="optional"/>
2154   <attribute name="AllowCreate" type="boolean" use="optional"/>
2155 </complexType>
```

2156 3.4.1.2 Element `<Scoping>`

2157 The `<Scoping>` element specifies the identity providers trusted by the requester to authenticate the  
 2158 presenter, as well as limitations and context related to proxying of the `<AuthnRequest>` message to  
 2159 subsequent identity providers by the responder. Its **ScopingType** complex type defines the following  
 2160 elements and attribute:

## 2161 ProxyCount [Optional]

2162 Specifies the number of proxying indirections permissible between the identity provider that receives  
 2163 this `<AuthnRequest>` and the identity provider who ultimately authenticates the principal. A count of  
 2164 zero permits no proxying, while omitting this attribute expresses no such restriction.

2165 <IDPList> [Optional]

2166 An advisory list of identity providers and associated information that the requester deems acceptable  
2167 to respond to the request.

2168 <RequesterID> [Zero or More]

2169 Identifies the set of requesting entities on whose behalf the requester is acting. Used to communicate  
2170 the chain of requesters when proxying occurs, as described in Section 3.4.1.5. See Section 8.3.6 for a  
2171 description of entity identifiers.

2172 In profiles specifying an active intermediary, the intermediary MAY examine the list and return a  
2173 <Response> message with an error <Status> and a second-level <StatusCode> of  
2174 urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP or  
2175 urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP if it cannot contact or does not support  
2176 any of the specified identity providers.

2177 The following schema fragment defines the <Scoping> element and its **ScopingType** complex type:

```
2178 <element name="Scoping" type="samlp:ScopingType"/>
2179 <complexType name="ScopingType">
2180   <sequence>
2181     <element ref="samlp:IDPList" minOccurs="0"/>
2182     <element ref="samlp:RequesterID" minOccurs="0" maxOccurs="unbounded"/>
2183   </sequence>
2184   <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
2185 </complexType>
2186 <element name="RequesterID" type="anyURI"/>
```

### 2187 3.4.1.3 Element <IDPList>

2188 The <IDPList> element specifies the identity providers trusted by the requester to authenticate the  
2189 presenter. Its **IDPListType** complex type defines the following elements:

2190 <IDPEntry> [One or More]

2191 Information about a single identity provider.

2192 <GetComplete> [Optional]

2193 If the <IDPList> is not complete, using this element specifies a URI reference that can be used to  
2194 retrieve the complete list. Retrieving the resource associated with the URI MUST result in an XML  
2195 instance whose root element is an <IDPList> that does not itself contain a <GetComplete>  
2196 element.

2197 The following schema fragment defines the <IDPList> element and its **IDPListType** complex type:

```
2198 <element name="IDPList" type="samlp:IDPListType"/>
2199 <complexType name="IDPListType">
2200   <sequence>
2201     <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
2202     <element ref="samlp:GetComplete" minOccurs="0"/>
2203   </sequence>
2204 </complexType>
2205 <element name="GetComplete" type="anyURI"/>
```

#### 2206 3.4.1.3.1 Element <IDPEntry>

2207 The <IDPEntry> element specifies a single identity provider trusted by the requester to authenticate the  
2208 presenter. Its **IDPEntryType** complex type defines the following attributes:

2209 ProviderID [Required]

2210 The unique identifier of the identity provider. See Section 8.3.6 for a description of such identifiers.

2211 Name [Optional]

2212 A human-readable name for the identity provider.

2213 Loc [Optional]

2214 A URI reference representing the location of a profile-specific endpoint supporting the authentication  
2215 request protocol. The binding to be used must be understood from the profile of use.

2216 The following schema fragment defines the <IDPEntry> element and its **IDPEntryType** complex type:

```
2217 <element name="IDPEntry" type="samlp:IDPEntryType"/>
2218 <complexType name="IDPEntryType">
2219   <attribute name="ProviderID" type="anyURI" use="required"/>
2220   <attribute name="Name" type="string" use="optional"/>
2221   <attribute name="Loc" type="anyURI" use="optional"/>
2222 </complexType>
```

### 2223 3.4.1.4 Processing Rules

2224 The <AuthnRequest> and <Response> exchange supports a variety of usage scenarios and is  
2225 therefore typically profiled for use in a specific context in which this optionality is constrained and specific  
2226 kinds of input and output are required or prohibited. The following processing rules apply as invariant  
2227 behavior across any profile of this protocol exchange. All other processing rules associated with the  
2228 underlying request and response messages **MUST** also be observed.

2229 The responder **MUST** ultimately reply to an <AuthnRequest> with a <Response> message containing  
2230 one or more assertions that meet the specifications defined by the request, or with a <Response>  
2231 message containing a <Status> describing the error that occurred. The responder **MAY** conduct  
2232 additional message exchanges with the presenter as needed to initiate or complete the authentication  
2233 process, subject to the nature of the protocol binding and the authentication mechanism. As described in  
2234 the next section, this includes proxying the request by directing the presenter to another identity provider  
2235 by issuing its own <AuthnRequest> message, so that the resulting assertion can be used to  
2236 authenticate the presenter to the original responder, in effect using SAML as the authentication  
2237 mechanism.

2238 If the responder is unable to authenticate the presenter or does not recognize the requested subject, or if  
2239 prevented from providing an assertion by policies in effect at the identity provider (for example the  
2240 intended subject has prohibited the identity provider from providing assertions to the relying party), then it  
2241 **MUST** return a <Response> with an error <Status>, and **MAY** return a second-level <StatusCode> of  
2242 urn:oasis:names:tc:SAML:2.0:status:AuthnFailed or  
2243 urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

2244 If the <saml:Subject> element in the request is present, then the resulting assertions'  
2245 <saml:Subject> **MUST strongly match** the request <saml:Subject>, as described in Section 3.3.4,  
2246 except that the identifier **MAY** be in a different format if specified by <NameIDPolicy>. In such a case,  
2247 the identifier's physical content **MAY** be different, but it **MUST** refer to the same principal.

2248 All of the content defined specifically within <AuthnRequest> is optional, although some may be required  
2249 by certain profiles. In the absence of any specific content at all, the following behavior is implied:

- 2250 • The assertion(s) returned **MUST** contain a <saml:Subject> element that represents the  
2251 presenter. The identifier type and format are determined by the identity provider. At least one  
2252 statement in at least one assertion **MUST** be a <saml:AuthnStatement> that describes the  
2253 authentication performed by the responder or authentication service associated with it.
- 2254 • The request presenter should, to the extent possible, be the only attesting entity able to satisfy the  
2255 <saml:SubjectConfirmation> of the assertion(s). In the case of weaker confirmation  
2256 methods, binding-specific or other mechanisms will be used to help satisfy this requirement.

- The resulting assertion(s) MUST contain a `<saml:AudienceRestriction>` element referencing the requester as an acceptable relying party. Other audiences MAY be included as deemed appropriate by the identity provider.

### 3.4.1.5 Proxying

If an identity provider that receives an `<AuthnRequest>` has not yet authenticated the presenter or cannot directly authenticate the presenter, but believes that the presenter has already authenticated to another identity provider or a non-SAML equivalent, it may respond to the request by issuing a new `<AuthnRequest>` on its own behalf to be presented to the other identity provider, or a request in whatever non-SAML format the entity recognizes. The original identity provider is termed the proxying identity provider.

Upon the successful return of a `<Response>` (or non-SAML equivalent) to the proxying provider, the enclosed assertion or non-SAML equivalent MAY be used to authenticate the presenter so that the proxying provider can issue an assertion of its own in response to the original `<AuthnRequest>`, completing the overall message exchange. Both the proxying and authenticating identity providers MAY include constraints on proxying activity in the messages and assertions they issue, as described in previous sections and below.

The requester can influence proxy behavior by including a `<Scoping>` element where the provider sets a desired `ProxyCount` value and/or indicates a list of preferred identity providers which may be proxied by including an ordered `<IDPList>` of preferred providers.

An identity provider can control secondary use of its assertions by proxying identity providers using a `<ProxyRestriction>` element in the assertions it issues.

#### 3.4.1.5.1 Proxying Processing Rules

An identity provider MAY proxy an `<AuthnRequest>` if the `<ProxyCount>` attribute is omitted or is greater than zero. Whether it chooses to proxy or not is a matter of local policy. An identity provider MAY choose to proxy for a provider specified in the `<IDPList>`, if provided, but is not required to do so.

An identity provider MUST NOT proxy a request where `<ProxyCount>` is set to zero. The identity provider MUST return an error `<Status>` containing a second-level `<StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded`, unless it can directly authenticate the presenter.

If it chooses to proxy to a SAML identity provider, when creating the new `<AuthnRequest>`, the proxying identity provider MUST include equivalent or stricter forms of all the information included in the original request (such as authentication context policy). Note, however, that the proxying provider is free to specify whatever `<NameIDPolicy>` it wishes to maximize the chances of a successful response.

If the authenticating identity provider is not a SAML identity provider, then the proxying provider MUST have some other way to ensure that the elements governing user agent interaction (`<IsPassive>`, for example) will be honored by the authenticating provider.

The new `<AuthnRequest>` MUST contain a `<ProxyCount>` attribute with a value of at most one less than the original value. If the original request does not contain a `<ProxyCount>` attribute, then the new request SHOULD contain a `<ProxyCount>` attribute.

If an `<IDPList>` was specified in the original request, the new request MUST also contain an `<IDPList>`. The proxying identity provider MAY add additional identity providers to the end of the `<IDPList>`, but MUST NOT remove any from the list.

2299 The authentication request and response are processed in normal fashion, in accordance with the rules  
 2300 given in this section and the profile of use. Once the presenter has authenticated to the proxying identity  
 2301 provider (in the case of SAML by delivering a <Response>), the following steps are followed:

- 2302 • The proxying identity provider prepares a new assertion on its own behalf by copying in the  
 2303 relevant information from the original assertion or non-SAML equivalent.
- 2304 • The new assertion's <saml:Subject> MUST contain an identifier that satisfies the original  
 2305 requester's preferences, as defined by its <NameIDPolicy> element.
- 2306 • The <saml:AuthnStatement> in the new assertion MUST include a <saml:AuthnContext>  
 2307 element containing a <saml:AuthenticatingAuthority> element referencing the identity  
 2308 provider to which the proxying identity provider referred the presenter. If the original assertion  
 2309 contains <saml:AuthnContext> information that includes one or more  
 2310 <saml:AuthenticatingAuthority> elements, those elements SHOULD be included in the  
 2311 new assertion, with the new element placed after them.
- 2312 • If the authenticating identity provider is not a SAML provider, then the proxying identity provider  
 2313 MUST generate a unique identifier value for the authenticating provider. This value SHOULD be  
 2314 consistent over time across different requests. The value MUST not conflict with values used or  
 2315 generated by other SAML providers.
- 2316 • Any other <saml:AuthnContext> information MAY be copied, translated, or omitted in  
 2317 accordance with the policies of the proxying identity provider, provided that the original  
 2318 requirements dictated by the requester are met.

2319 If, in the future, the identity provider is asked to authenticate the same presenter for a second requester,  
 2320 and this request is equally or less strict than the original request (as determined by the proxying identity  
 2321 provider), the identity provider MAY skip the creation of a new <AuthnRequest> to the authenticating  
 2322 identity provider and immediately issue another assertion (assuming the original assertion or non-SAML  
 2323 equivalent it received is still valid).

### 2324 **3.5 Artifact Resolution Protocol**

2325 The artifact resolution protocol provides a mechanism by which SAML protocol messages can be  
 2326 transported in a SAML binding by reference instead of by value. Both requests and responses can be  
 2327 obtained by reference using this specialized protocol. A message sender, instead of binding a message to  
 2328 a transport protocol, sends a small piece of data called an artifact using the binding. An artifact can take a  
 2329 variety of forms, but must support a means by which the receiver can determine who sent it. If the receiver  
 2330 wishes, it can then use this protocol in conjunction with a different (generally synchronous) SAML binding  
 2331 protocol to resolve the artifact into the original protocol message.

2332 The most common use for this mechanism is with bindings that cannot easily carry a message because of  
 2333 size constraints, or to enable a message to be communicated via a secure channel between the SAML  
 2334 requester and responder, avoiding the need for a signature.

2335 Depending on the characteristics of the underlying message being passed by reference, the artifact  
 2336 resolution protocol MAY require protections such as mutual authentication, integrity protection,  
 2337 confidentiality, etc. from the protocol binding used to resolve the artifact. In all cases, the artifact MUST  
 2338 exhibit a single-use semantic such that once it has been successfully resolved, it can no longer be used  
 2339 by any party.

2340 Regardless of the protocol message obtained, the result of resolving an artifact MUST be treated exactly  
 2341 as if the message so obtained had been sent originally in place of the artifact.



### 3.5.1 Element <ArtifactResolve>

The <ArtifactResolve> message is used to request that a SAML protocol message be returned in an <ArtifactResponse> message by specifying an artifact that represents the SAML protocol message. The original transmission of the artifact is governed by the specific protocol binding that is being used; see [SAMLBind] for more information on the use of artifacts in bindings.

The <ArtifactResolve> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ArtifactResolveType**, which extends **RequestAbstractType** and adds the following element:

<Artifact> [Required]

The artifact value that the requester received and now wishes to translate into the protocol message it represents. See [SAMLBind] for specific artifact format information.

The following schema fragment defines the <ArtifactResolve> element and its **ArtifactResolveType** complex type:

```
<element name="ArtifactResolve" type="samlp:ArtifactResolveType"/>
<complexType name="ArtifactResolveType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <element ref="samlp:Artifact"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="Artifact" type="string"/>
```

### 3.5.2 Element <ArtifactResponse>

The recipient of an <ArtifactResolve> message MUST respond with an <ArtifactResponse> message element. This element is of complex type **ArtifactResponseType**, which extends **StatusResponseType** with a single optional wildcard element corresponding to the SAML protocol message being returned. This wrapped message element can be a request or a response.

The <ArtifactResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ArtifactResponse> element and its **ArtifactResponseType** complex type:

```
<element name="ArtifactResponse" type="samlp:ArtifactResponseType"/>
<complexType name="ArtifactResponseType">
  <complexContent>
    <extension base="samlp:StatusResponseType">
      <sequence>
        <any namespace="##any" processContents="lax" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

### 3.5.3 Processing Rules

If the responder recognizes the artifact as valid, then it responds with the associated protocol message in an <ArtifactResponse> message element. Otherwise, it responds with an <ArtifactResponse>



element with no embedded message. In both cases, the `<Status>` element MUST include a `<StatusCode>` element with the code value `urn:oasis:names:tc:SAML:2.0:status:Success`. A response message with no embedded message inside it is termed an empty response in the remainder of this section.

The responder MUST enforce a one-time-use property on the artifact by ensuring that any subsequent request with the same artifact by any requester results in an empty response as described above.

Some SAML protocol messages, most particularly the `<AuthnRequest>` message in some profiles, MAY be intended for consumption by any party that receives it and can respond appropriately. In most other cases, however, a message is intended for a specific entity. In such cases, the artifact when issued MUST be associated with the intended recipient of the message that the artifact represents. If the artifact issuer receives an `<ArtifactResolve>` message from a requester that cannot authenticate itself as the original intended recipient, then the artifact issuer MUST return an empty response.

The artifact issuer SHOULD enforce the shortest practical time limit on the usability of an artifact, such that an acceptable window of time (but no more) exists for the artifact receiver to obtain the artifact and return it in an `<ArtifactResolve>` message to the issuer.

Note that the `<ArtifactResponse>` message's `InResponseTo` attribute MUST contain the value of the corresponding `<ArtifactResolve>` message's `ID` attribute, but the embedded protocol message will contain its own message identifier, and in the case of an embedded response, may contain a different `InResponseTo` value that corresponds to the original request message to which the embedded message is responding.

All other processing rules associated with the underlying request and response messages MUST be observed.

## 3.6 Name Identifier Management Protocol

After establishing a name identifier for a principal, an identity provider wishing to change the value and/or format of the identifier that it will use when referring to the principal, or to indicate that a name identifier will no longer be used to refer to the principal, informs service providers of the change by sending them a `<ManageNameIDRequest>` message.

A service provider also uses this message to register or change the `SPProvidedID` value to be included when the underlying name identifier is used to communicate with it, or to terminate the use of a name identifier between itself and the identity provider.

Note that this protocol is typically not used with "transient" name identifiers, since their value is not intended to be managed on a long term basis.

### 3.6.1 Element `<ManageNameIDRequest>`

A provider sends a `<ManageNameIDRequest>` message to inform the recipient of a changed name identifier or to indicate the termination of the use of a name identifier.

The `<ManageNameIDRequest>` message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **ManageNameIDRequestType**, which extends **RequestAbstractType** and adds the following elements:

`<saml:NameID>` or `<saml:EncryptedID>` [Required]

The name identifier and associated descriptive data (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request. (For more information on these elements, see Section 2.2.)

<NewID> or <NewEncryptedID> or <Terminate> [Required]

The new identifier value (in plaintext or encrypted form) to be used when communicating with the requesting provider concerning this principal, or an indication that the use of the old identifier has been terminated. In the former case, if the requester is the service provider, the new identifier MUST appear in subsequent <NameID> elements in the SPProvidedID attribute. If the requester is the identity provider, the new value will appear in subsequent <NameID> elements as the element's content.

The following schema fragment defines the <ManageNameIDRequest> element and its **ManageNameIDRequestType** complex type:

```
<element name="ManageNameIDRequest" type="samlp:ManageNameIDRequestType"/>
<complexType name="ManageNameIDRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <choice>
          <element ref="saml:NameID"/>
          <element ref="saml:EncryptedID"/>
        </choice>
        <choice>
          <element ref="samlp:NewID"/>
          <element ref="samlp:NewEncryptedID"/>
          <element ref="samlp:Terminate"/>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
<element name="NewID" type="string"/>
<element name="NewEncryptedID" type="saml:EncryptedElementType"/>
<element name="Terminate" type="samlp:TerminateType"/>
<complexType name="TerminateType"/>
```

### 3.6.2 Element <ManageNameIDResponse>

The recipient of a <ManageNameIDRequest> message MUST respond with a <ManageNameIDResponse> message, which is of type **StatusResponseType** with no additional content.

The <ManageNameIDResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <ManageNameIDResponse> element:

```
<element name="ManageNameIDResponse" type="samlp:StatusResponseType"/>
```

### 3.6.3 Processing Rules

If the request includes a <saml:NameID> (or encrypted version) that the recipient does not recognize, the responding provider MUST respond with an error <Status> and MAY respond with a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

If the <Terminate> element is included in the request, the requesting provider is indicating that (in the case of a service provider) it will no longer accept assertions from the identity provider or (in the case of an identity provider) it will no longer issue assertions to the service provider about the principal. The receiving provider can perform any maintenance with the knowledge that the relationship represented by the name identifier has been terminated. It can choose to invalidate the active session(s) of a principal for whom a relationship has been terminated.

2481 If the service provider requests that its identifier for the principal be changed by including a `<NewID>` (or  
 2482 `<NewEncryptedID>`) element, the identity provider MUST include the element's content as the  
 2483 `SPProvidedID` when subsequently communicating to the service provider regarding this principal.

2484 If the identity provider requests that its identifier for the principal be changed by including a `<NewID>` (or  
 2485 `<NewEncryptedID>`) element, the service provider MUST use the element's content as the  
 2486 `<saml:NameID>` element content when subsequently communicating with the identity provider regarding  
 2487 this principal.

2488 Note that neither, either, or both of the original and new identifier MAY be encrypted (using the  
 2489 `<EncryptedID>` and `<NewEncryptedID>` elements).

2490 In any case, the `<saml:NameID>` content in the request and its associated `SPProvidedID` attribute  
 2491 MUST contain the most recent name identifier information established between the providers for the  
 2492 principal.

2493 In the case of an identifier with a `Format` of `urn:oasis:names:tc:SAML:2.0:nameid-`  
 2494 `format:persistent`, the `NameQualifier` attribute MUST contain the unique identifier of the identity  
 2495 provider that created the identifier. If the identifier was established between the identity provider and an  
 2496 affiliation group of which the service provider is a member, then the `SPNameQualifier` attribute MUST  
 2497 contain the unique identifier of the affiliation group. Otherwise, it MUST contain the unique identifier of the  
 2498 service provider. These attributes MAY be omitted if they would otherwise match the value of the  
 2499 containing protocol message's `<Issuer>` element, but this is NOT RECOMMENDED due to the  
 2500 opportunity for confusion.

2501 Changes to these identifiers may take a potentially significant amount of time to propagate through the  
 2502 systems at both the requester and the responder. Implementations might wish to allow each party to  
 2503 accept either identifier for some period of time following the successful completion of a name identifier  
 2504 change. Not doing so could result in the inability of the principal to access resources.

2505 All other processing rules associated with the underlying request and response messages MUST be  
 2506 observed.

### 2507 **3.7 Single Logout Protocol**

2508 The single logout protocol provides a message exchange protocol by which all sessions provided by a  
 2509 particular session authority are near-simultaneously terminated. The single logout protocol is used either  
 2510 when a principal logs out at a session participant or when the principal logs out directly at the  
 2511 session authority. This protocol may also be used to log out a principal due to a timeout. The reason for  
 2512 the logout event can be indicated through the `Reason` attribute.

2513  
 2514 The principal may have established authenticated sessions with both the session authority and individual  
 2515 session participants, based on assertions containing authentication statements supplied by the session  
 2516 authority.

2517  
 2518 When the principal invokes the single logout process at a session participant, the session participant  
 2519 MUST send a `<LogoutRequest>` message to the session authority that provided the assertion  
 2520 containing the authentication statement related to that session at the session participant.

2521  
 2522 When either the principal invokes a logout at the session authority, or a session participant sends a logout  
 2523 request to the session authority specifying that principal, the session authority SHOULD send a  
 2524 `<LogoutRequest>` message to each session participant to which it provided assertions containing  
 2525 authentication statements under its current session with the principal, with the exception of the session  
 2526 participant that sent the `<LogoutRequest>` message to the session authority. It SHOULD attempt to  
 2527 contact as many of these participants as it can using this protocol, terminate its own session with the  
 2528 principal, and finally return a `<LogoutResponse>` message to the requesting session participant, if any.

### 3.7.1 Element <LogoutRequest>

A session participant or session authority sends a <LogoutRequest> message to indicate that a session has been terminated.

The <LogoutRequest> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **LogoutRequestType**, which extends **RequestAbstractType** and adds the following elements and attributes:

**NotOnOrAfter** [Optional]

The time at which the request expires, after which the recipient may discard the message. The time value is encoded in UTC, as described in Section 1.3.3.

**Reason** [Optional]

An indication of the reason for the logout, in the form of a URI reference.

<saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

The identifier and associated attributes (in plaintext or encrypted form) that specify the principal as currently recognized by the identity and service providers prior to this request. (For more information on this element, see Section 2.2.)

<SessionIndex> [Optional]

The identifier that indexes this session at the message recipient.

The following schema fragment defines the <LogoutRequest> element and associated **LogoutRequestType** complex type:

```
<element name="LogoutRequest" type="samlp:LogoutRequestType"/>
<complexType name="LogoutRequestType">
  <complexContent>
    <extension base="samlp:RequestAbstractType">
      <sequence>
        <choice>
          <element ref="saml:BaseID"/>
          <element ref="saml:NameID"/>
          <element ref="saml:EncryptedID"/>
        </choice>
        <element ref="samlp:SessionIndex" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="Reason" type="string" use="optional"/>
      <attribute name="NotOnOrAfter" type="dateTime"
use="optional"/>
    </extension>
  </complexContent>
</complexType>
<element name="SessionIndex" type="string"/>
```

### 3.7.2 Element <LogoutResponse>

The recipient of a <LogoutRequest> message MUST respond with a <LogoutResponse> message, of type **StatusResponseType**, with no additional content specified.

The <LogoutResponse> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <LogoutResponse> element:

2575 `<element name="LogoutResponse" type="samlp:StatusResponseType"/>`

### 2576 3.7.3 Processing Rules

2577 The message sender MAY use the `Reason` attribute to indicate the reason for sending the  
 2578 `<LogoutRequest>`. The following values are defined by this specification for use by all message  
 2579 senders; other values MAY be agreed on between participants:

2580 `urn:oasis:names:tc:SAML:2.0:logout:user`

2581 Specifies that the message is being sent because the principal wishes to terminate the indicated  
 2582 session.

2583 `urn:oasis:names:tc:SAML:2.0:logout:admin`

2584 Specifies that the message is being sent because an administrator wishes to terminate the indicated  
 2585 session for that principal.

2586 All other processing rules associated with the underlying request and response messages MUST be  
 2587 observed.

2588 Additional processing rules are provided in the following sections.

#### 2589 3.7.3.1 Session Participant Rules

2590 When a session participant receives a `<LogoutRequest>` message, the session participant MUST  
 2591 authenticate the message. If the sender is the authority that provided an assertion containing an  
 2592 authentication statement linked to the principal's current session, the session participant MUST invalidate  
 2593 the principal's session(s) referred to by the `<saml:BaseID>`, `<saml:NameID>`, or  
 2594 `<saml:EncryptedID>` element, and any `<SessionIndex>` elements supplied in the message. If no  
 2595 `<SessionIndex>` elements are supplied, then all sessions associated with the principal MUST be  
 2596 invalidated.

2597  
 2598 The session participant MUST apply the logout request message to any assertion that meets the following  
 2599 conditions, even if the assertion arrives after the logout request:

- 2600 • The subject of the assertion **strongly matches** the `<saml:BaseID>`, `<saml:NameID>`, or  
 2601 `<saml:EncryptedID>` element in the `<LogoutRequest>`, as defined in Section 3.3.4.
- 2602 • The `SessionIndex` attribute of one of the assertion's authentication statements matches one of  
 2603 the `<SessionIndex>` elements specified in the logout request, or the logout request contains no  
 2604 `<SessionIndex>` elements.
- 2605 • The assertion would otherwise be valid, based on the time conditions specified in the assertion itself  
 2606 (in particular, the value of any specified `NotOnOrAfter` attributes in conditions or subject  
 2607 confirmation data).
- 2608 • The logout request has not yet expired (determined by examining the `NotOnOrAfter` attribute on  
 2609 the message).

2610 **Note:** This rule is intended to prevent a situation in which a session participant receives a  
 2611 logout request targeted at a single, or multiple, assertion(s) (as identified by the  
 2612 `<SessionIndex>` element(s)) *before* it receives the actual – and possibly still valid –  
 2613 assertion(s) targeted by the logout request. It should honor the logout request until the  
 2614 logout request itself may be discarded (the `NotOnOrAfter` value on the request has  
 2615 been exceeded) or the assertion targeted by the logout request has been received and  
 2616 has been handled appropriately.

### 3.7.3.2 Session Authority Rules

When a session authority receives a `<LogoutRequest>` message, the session authority **MUST** authenticate the sender. If the sender is a session participant to which the session authority provided an assertion containing an authentication statement for the current session, then the session authority **SHOULD** do the following in the specified order:

- Send a `<LogoutRequest>` message to any session authority on behalf of whom the session authority proxied the principal's authentication, unless the second authority is the originator of the `<LogoutRequest>`.
- Send a `<LogoutRequest>` message to each session participant for which the session authority provided assertions in the current session, *other than* the originator of a current `<LogoutRequest>`.
- Terminate the principal's current session as specified by the `<saml:BaseID>`, `<saml:NameID>`, or `<saml:EncryptedID>` element, and any `<SessionIndex>` elements present in the logout request message.

If the session authority successfully terminates the principal's session with respect to itself, then it **MUST** respond to the original requester, if any, with a `<LogoutResponse>` message containing a top-level status code of `urn:oasis:names:tc:SAML:2.0:status:Success`. If it cannot do so, then it **MUST** respond with a `<LogoutResponse>` message containing a top-level status code indicating the error. Thus, the top-level status indicates the state of the logout operation only with respect to the session authority itself.

The session authority **SHOULD** attempt to contact each session participant using any applicable/usable protocol binding, even if one or more of these attempts fails or cannot be attempted (for example because the original request takes place using a protocol binding that does not enable the logout to be propagated to all participants).

In the event that not all session participants successfully respond to these `<LogoutRequest>` messages (or if not all participants can be contacted), then the session authority **MUST** include in its `<LogoutResponse>` message a second-level status code of `urn:oasis:names:tc:SAML:2.0:status:PartialLogout` to indicate that not all other session participants successfully responded with confirmation of the logout.

Note that a session authority **MAY** initiate a logout for reasons other than having received a `<LogoutRequest>` from a session participant – these include, but are not limited to:

- If some timeout period was agreed out-of-band with an individual session participant, the session authority **MAY** send a `<LogoutRequest>` to that individual participant alone.
- An agreed global timeout period has been exceeded.
- The principal or some other trusted entity has requested logout of the principal directly at the session authority.
- The session authority has determined that the principal's credentials may have been compromised.

When constructing a logout request message, the session authority **MUST** set the value of the `NotOnOrAfter` attribute of the message to a time value, indicating an expiration time for the message, after which the logout request may be discarded by the recipient. This value **SHOULD** be set to a time value equal to or greater than the value of any `NotOnOrAfter` attribute specified in the assertion most recently issued as part of the targeted session (as indicated by the `SessionIndex` attribute on the logout request).

In addition to the values specified in Section 3.6.3 for the `Reason` attribute, the following values are also available for use by the session authority only:

`urn:oasis:names:tc:SAML:2.0:logout:global-timeout`



2663 Specifies that the message is being sent because of the global session timeout interval period  
2664 being exceeded.

2665 urn:oasis:names:tc:SAML:2.0:logout:sp-timeout

2666 Specifies that the message is being sent because a timeout interval period agreed between a  
2667 participant and the session authority has been exceeded.

## 2668 3.8 Name Identifier Mapping Protocol

2669 When an entity that shares an identifier for a principal with an identity provider wishes to obtain a name  
2670 identifier for the same principal in a particular format or federation namespace, it can send a request to  
2671 the identity provider using this protocol.

2672 For example, a service provider that wishes to communicate with another service provider with whom it  
2673 does not share an identifier for the principal can use an identity provider that shares an identifier for the  
2674 principal with both service providers to map from its own identifier to a new identifier, generally encrypted,  
2675 with which it can communicate with the second service provider.

2676 Regardless of the type of identifier involved, the mapped identifier SHOULD be encrypted into a  
2677 <saml:EncryptedID> element unless a specific deployment dictates such protection is unnecessary.

### 2678 3.8.1 Element <NameIDMappingRequest>

2679 To request an alternate name identifier for a principal from an identity provider, a requester sends an  
2680 <NameIDMappingRequest> message. This message has the complex type  
2681 **NameIDMappingRequestType**, which extends **RequestAbstractType** and adds the following elements:

2682 <saml:BaseID> or <saml:NameID> or <saml:EncryptedID> [Required]

2683 The identifier and associated descriptive data that specify the principal as currently recognized by the  
2684 requester and the responder. (For more information on this element, see Section 2.2.)

2685 <NameIDPolicy> [Required]

2686 The requirements regarding the format and optional name qualifier for the identifier to be returned.

2687 The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol  
2688 binding used to deliver the message.

2689 The following schema fragment defines the <NameIDMappingRequest> element and its  
2690 **NameIDMappingRequestType** complex type:

```
2691 <element name="NameIDMappingRequest" type="samlp:NameIDMappingRequestType"/>
2692 <complexType name="NameIDMappingRequestType">
2693   <complexContent>
2694     <extension base="samlp:RequestAbstractType">
2695       <sequence>
2696         <choice>
2697           <element ref="saml:BaseID"/>
2698           <element ref="saml:NameID"/>
2699           <element ref="saml:EncryptedID"/>
2700         </choice>
2701         <element ref="samlp:NameIDPolicy"/>
2702       </sequence>
2703     </extension>
2704   </complexContent>
2705 </complexType>
```

### 3.8.2 Element <NameIDMappingResponse>

The recipient of a <NameIDMappingRequest> message MUST respond with a <NameIDMappingResponse> message. This message has the complex type **NameIDMappingResponseType**, which extends **StatusResponseType** and adds the following element:

<saml:NameID> or <saml:EncryptedID> [Required]

The identifier and associated attributes that specify the principal in the manner requested, usually in encrypted form. (For more information on this element, see Section 2.2.)

The message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

The following schema fragment defines the <NameIDMappingResponse> element and its **NameIDMappingResponseType** complex type:

```
<element name="NameIDMappingResponse" type="samlp:NameIDMappingResponseType"/>
<complexType name="NameIDMappingResponseType">
  <complexContent>
    <extension base="samlp:StatusResponseType">
      <choice>
        <element ref="saml:NameID"/>
        <element ref="saml:EncryptedID"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

### 3.8.3 Processing Rules

If the responder does not recognize the principal identified in the request, it MAY respond with an error <Status> containing a second-level <StatusCode> of urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

At the responder's discretion, the urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy status code MAY be returned to indicate an inability or unwillingness to supply an identifier in the requested format or namespace.

All other processing rules associated with the underlying request and response messages MUST be observed.

## 4 SAML Versioning

The SAML specification set is versioned in two independent ways. Each is discussed in the following sections, along with processing rules for detecting and handling version differences. Also included are guidelines on when and why specific version information is expected to change in future revisions of the specification.

When version information is expressed as both a Major and Minor version, it is expressed in the form *Major.Minor*. The version number *Major<sub>B</sub>.Minor<sub>B</sub>* is higher than the version number *Major<sub>A</sub>.Minor<sub>A</sub>* if and only if:

$(Major_B > Major_A) \text{ OR } ( (Major_B = Major_A) \text{ AND } (Minor_B > Minor_A) )$

### 4.1 SAML Specification Set Version

Each release of the SAML specification set will contain a major and minor version designation describing its relationship to earlier and later versions of the specification set. The version will be expressed in the content and filenames of published materials, including the specification set documents and XML schema documents. There are no normative processing rules surrounding specification set versioning, since it merely encompasses the collective release of normative specification documents which themselves contain processing rules.

The overall size and scope of changes to the specification set documents will informally dictate whether a set of changes constitutes a major or minor revision. In general, if the specification set is backwards compatible with an earlier specification set (that is, valid older syntax, protocols, and semantics remain valid), then the new version will be a minor revision. Otherwise, the changes will constitute a major revision.

#### 4.1.1 Schema Version

As a non-normative documentation mechanism, any XML schema documents published as part of the specification set will contain a `version` attribute on the `<xs:schema>` element whose value is in the form *Major.Minor*, reflecting the specification set version in which it has been published. Validating implementations MAY use the attribute as a means of distinguishing which version of a schema is being used to validate messages, or to support multiple versions of the same logical schema.

#### 4.1.2 SAML Assertion Version

The SAML `<Assertion>` element contains an attribute for expressing the major and minor version of the assertion in a string of the form *Major.Minor*. Each version of the SAML specification set will be construed so as to document the syntax, semantics, and processing rules of the assertions of the same version. That is, specification set version 1.0 describes assertion version 1.0, and so on.

There is explicitly NO relationship between the assertion version and the target XML namespace specified for the schema definitions for that assertion version.

The following processing rules apply:

- A SAML asserting party MUST NOT issue any assertion with an overall *Major.Minor* assertion version number not supported by the authority.
- A SAML relying party MUST NOT process any assertion with a major assertion version number not supported by the relying party.
- A SAML relying party MAY process or MAY reject an assertion whose minor assertion version number is higher than the minor assertion version number supported by the relying party. However, all assertions that share a major assertion version number MUST share the same general

2779 processing rules and semantics, and MAY be treated in a uniform way by an implementation. For  
 2780 example, if a V1.1 assertion shares the syntax of a V1.0 assertion, an implementation MAY treat the  
 2781 assertion as a V1.0 assertion without ill effect. (See Section 4.2.1 for more information about the  
 2782 likely effects of schema evolution.)

### 2783 4.1.3 SAML Protocol Version

2784 The various SAML protocols' request and response elements contain an attribute for expressing the major  
 2785 and minor version of the request or response message using a string of the form *Major.Minor*. Each  
 2786 version of the SAML specification set will be construed so as to document the syntax, semantics, and  
 2787 processing rules of the protocol messages of the same version. That is, specification set version 1.0  
 2788 describes request and response version V1.0, and so on.

2789 There is explicitly NO relationship between the protocol version and the target XML namespace specified  
 2790 for the schema definitions for that protocol version.

2791 The version numbers used in SAML protocol request and response elements will match for any particular  
 2792 revision of the SAML specification set.

#### 2793 4.1.3.1 Request Version

2794 The following processing rules apply to requests:

- 2795 • A SAML requester SHOULD issue requests with the highest request version supported by both the  
 2796 SAML requester and the SAML responder.
- 2797 • If the SAML requester does not know the capabilities of the SAML responder, then it SHOULD  
 2798 assume that the responder supports requests with the highest request version supported by the  
 2799 requester.
- 2800 • A SAML requester MUST NOT issue a request message with an overall *Major.Minor* request version  
 2801 number matching a response version number that the requester does not support.
- 2802 • A SAML responder MUST reject any request with a major request version number not supported by  
 2803 the responder.
- 2804 • A SAML responder MAY process or MAY reject any request whose minor request version number is  
 2805 higher than the highest supported request version that it supports. However, all requests that share  
 2806 a major request version number MUST share the same general processing rules and semantics,  
 2807 and MAY be treated in a uniform way by an implementation. That is, if a V1.1 request shares the  
 2808 syntax of a V1.0 request, a responder MAY treat the request message as a V1.0 request without ill  
 2809 effect. (See Section 4.2.1 for more information about the likely effects of schema evolution.)

#### 2810 4.1.3.2 Response Version

2811 The following processing rules apply to responses:

- 2812 • A SAML responder MUST NOT issue a response message with a response version number higher  
 2813 than the request version number of the corresponding request message.
- 2814 • A SAML responder MUST NOT issue a response message with a major response version number  
 2815 lower than the major request version number of the corresponding request message except to  
 2816 report the error `urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh`.
- 2817 • An error response resulting from incompatible SAML protocol versions MUST result in reporting a  
 2818 top-level `<StatusCode>` value of  
 2819 `urn:oasis:names:tc:SAML:2.0:status:VersionMismatch`, and MAY result in reporting  
 2820 one of the following second-level values:

2821 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh,  
 2822 urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow, or  
 2823 urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated.

### 2824 4.1.3.3 Permissible Version Combinations

2825 Assertions of a particular major version appear only in response messages of the same major version, as  
 2826 permitted by the importation of the SAML assertion namespace into the SAML protocol schema. For  
 2827 example, a V1.1 assertion MAY appear in a V1.0 response message, and a V1.0 assertion in a V1.1  
 2828 response message, if the appropriate assertion schema is referenced during namespace importation. But  
 2829 a V1.0 assertion MUST NOT appear in a V2.0 response message because they are of different major  
 2830 versions.

## 2831 4.2 SAML Namespace Version

2832 XML schema documents published as part of the specification set contain one or more target  
 2833 namespaces into which the type, element, and attribute definitions are placed. Each namespace is distinct  
 2834 from the others, and represents, in shorthand, the structural and syntactic definitions that make up that  
 2835 part of the specification.

2836 The namespace URI references defined by the specification set will generally contain version information  
 2837 of the form *Major.Minor* somewhere in the URI. The major and minor version in the URI MUST correspond  
 2838 to the major and minor version of the specification set in which the namespace is first introduced and  
 2839 defined. This information is not typically consumed by an XML processor, which treats the namespace  
 2840 opaquely, but is intended to communicate the relationship between the specification set and the  
 2841 namespaces it defines. This pattern is also followed by the SAML-defined URI-based identifiers that are  
 2842 listed in Section 8.

2843 As a general rule, implementers can expect the namespaces and the associated schema definitions  
 2844 defined by a major revision of the specification set to remain valid and stable across minor revisions of the  
 2845 specification. New namespaces may be introduced, and when necessary, old namespaces replaced, but  
 2846 this is expected to be rare. In such cases, the older namespaces and their associated definitions should  
 2847 be expected to remain valid until a major specification set revision.

### 2848 4.2.1 Schema Evolution

2849 In general, maintaining namespace stability while adding or changing the content of a schema are  
 2850 competing goals. While certain design strategies can facilitate such changes, it is complex to predict how  
 2851 older implementations will react to any given change, making forward compatibility difficult to achieve.  
 2852 Nevertheless, the right to make such changes in minor revisions is reserved, in the interest of namespace  
 2853 stability. Except in special circumstances (for example, to correct major deficiencies or to fix errors),  
 2854 implementations should expect forward-compatible schema changes in minor revisions, allowing new  
 2855 messages to validate against older schemas.

2856 Implementations SHOULD expect and be prepared to deal with new extensions and message types in  
 2857 accordance with the processing rules laid out for those types. Minor revisions MAY introduce new types  
 2858 that leverage the extension facilities described in Section 7. Older implementations SHOULD reject such  
 2859 extensions gracefully when they are encountered in contexts that dictate mandatory semantics. Examples  
 2860 include new query, statement, or condition types.

## 5 SAML and XML Signature Syntax and Processing

SAML assertions and SAML protocol request and response messages may be signed, with the following benefits. An assertion signed by the asserting party supports assertion integrity, authentication of the asserting party to a SAML relying party, and, if the signature is based on the SAML authority's public-private key pair, non-repudiation of origin. A SAML protocol request or response message signed by the message originator supports message integrity, authentication of message origin to a destination, and, if the signature is based on the originator's public-private key pair, non-repudiation of origin.

A digital signature is not always required in SAML. For example, in some circumstances, signatures may be "inherited," such as when an unsigned assertion gains protection from a signature on the containing protocol response message. "Inherited" signatures should be used with care when the contained object (such as the assertion) is intended to have a non-transitory lifetime. The reason is that the entire context must be retained to allow validation, exposing the XML content and adding potentially unnecessary overhead. As another example, the SAML relying party or SAML requester may have obtained an assertion or protocol message from the SAML asserting party or SAML responder directly (with no intermediaries) through a secure channel, with the asserting party or SAML responder having authenticated to the relying party or SAML responder by some means other than a digital signature.

Many different techniques are available for "direct" authentication and secure channel establishment between two parties. The list includes TLS/SSL (see [RFC 2246]/[SSL3]), HMAC, password-based mechanisms, and so on. In addition, the applicable security requirements depend on the communicating applications and the nature of the assertion or message transported. It is RECOMMENDED that, in all other contexts, digital signatures be used for assertions and request and response messages. Specifically:

- A SAML assertion obtained by a SAML relying party from an entity other than the SAML asserting party SHOULD be signed by the SAML asserting party.
- A SAML protocol message arriving at a destination from an entity other than the originating sender SHOULD be signed by the sender.
- Profiles MAY specify alternative signature mechanisms such as S/MIME or signed Java objects that contain SAML documents. Caveats about retaining context and interoperability apply. XML Signatures are intended to be the primary SAML signature mechanism, but this specification attempts to ensure compatibility with profiles that may require other mechanisms.
- Unless a profile specifies an alternative signature mechanism, any XML Digital Signatures MUST be enveloped.

### 5.1 Signing Assertions

All SAML assertions MAY be signed using XML Signature. This is reflected in the assertion schema as described in Section 2.

### 5.2 Request/Response Signing

All SAML protocol request and response messages MAY be signed using XML Signature. This is reflected in the schema as described in Section 3.

### 5.3 Signature Inheritance

A SAML assertion may be embedded within another SAML element, such as an enclosing `<Assertion>` or a request or response, which may be signed. When a SAML assertion does not contain a `<ds:Signature>` element, but is contained in an enclosing SAML element that contains a `<ds:Signature>` element, and the signature applies to the `<Assertion>` element and all its children,



2904 then the assertion can be considered to inherit the signature from the enclosing element. The resulting  
 2905 interpretation should be equivalent to the case where the assertion itself was signed with the same key  
 2906 and signature options.

2907 Many SAML use cases involve SAML XML data enclosed within other protected data structures such as  
 2908 signed SOAP messages, S/MIME packages, and authenticated SSL connections. SAML profiles MAY  
 2909 define additional rules for interpreting SAML elements as inheriting signatures or other authentication  
 2910 information from the surrounding context, but no such inheritance should be inferred unless specifically  
 2911 identified by the profile.

## 2912 **5.4 XML Signature Profile**

2913 The XML Signature specification [XMLSig] calls out a general XML syntax for signing data with flexibility  
 2914 and many choices. This section details constraints on these facilities so that SAML processors do not  
 2915 have to deal with the full generality of XML Signature processing. This usage makes specific use of the  
 2916 **xs:ID**-typed attributes present on the root elements to which signatures can apply, specifically the **ID**  
 2917 attribute on `<Assertion>` and the various request and response elements. These attributes are  
 2918 collectively referred to in this section as the identifier attributes.

2919 Note that this profile only applies to the use of the `<ds:Signature>` elements found directly within SAML  
 2920 assertions, requests, and responses. Other profiles in which signatures appear elsewhere but apply to  
 2921 SAML content are free to define other approaches.

### 2922 **5.4.1 Signing Formats and Algorithms**

2923 XML Signature has three ways of relating a signature to a document: enveloping, enveloped, and  
 2924 detached.

2925 SAML assertions and protocols MUST use enveloped signatures when signing assertions and protocol  
 2926 messages. SAML processors SHOULD support the use of RSA signing and verification for public key  
 2927 operations in accordance with the algorithm identified by <http://www.w3.org/2000/09/xmldsig#rsa-sha1>.

### 2928 **5.4.2 References**

2929 SAML assertions and protocol messages MUST supply a value for the **ID** attribute on the root element of  
 2930 the assertion or protocol message being signed. The assertion's or protocol message's root element may  
 2931 or may not be the root element of the actual XML document containing the signed assertion or protocol  
 2932 message (e.g., it might be contained within a SOAP envelope).

2933 Signatures MUST contain a single `<ds:Reference>` containing a same-document reference to the **ID**  
 2934 attribute value of the root element of the assertion or protocol message being signed. For example, if the  
 2935 **ID** attribute value is "foo", then the **URI** attribute in the `<ds:Reference>` element MUST be "#foo".

### 2936 **5.4.3 Canonicalization Method**

2937 SAML implementations SHOULD use Exclusive Canonicalization [Excl-C14N], with or without comments,  
 2938 both in the `<ds:CanonicalizationMethod>` element of `<ds:SignedInfo>`, and as a  
 2939 `<ds:Transform>` algorithm. Use of Exclusive Canonicalization ensures that signatures created over  
 2940 SAML messages embedded in an XML context can be verified independent of that context.

### 2941 **5.4.4 Transforms**

2942 Signatures in SAML messages SHOULD NOT contain transforms other than the enveloped signature  
 2943 transform (with the identifier <http://www.w3.org/2000/09/xmldsig#enveloped-signature>) or the exclusive

2944 canonicalization transforms (with the identifier <http://www.w3.org/2001/10/xml-exc-c14n#> or  
2945 <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>).

2946 Verifiers of signatures MAY reject signatures that contain other transform algorithms as invalid. If they do  
2947 not, verifiers MUST ensure that no content of the SAML message is excluded from the signature. This can  
2948 be accomplished by establishing out-of-band agreement as to what transforms are acceptable, or by  
2949 applying the transforms manually to the content and reverifying the result as consisting of the same SAML  
2950 message.

## 2951 5.4.5 KeyInfo

2952 XML Signature defines usage of the `<ds:KeyInfo>` element. SAML does not require the use of  
2953 `<ds:KeyInfo>`, nor does it impose any restrictions on its use. Therefore, `<ds:KeyInfo>` MAY be  
2954 absent.

## 2955 5.4.6 Example

2956 Following is an example of a signed response containing a signed assertion. Line breaks have been  
2957 added for readability; the signatures are not valid and cannot be successfully verified.

```
2958 <Response
2959   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
2960   ID="_c7055387-af61-4fce-8b98-e2927324b306"
2961   xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
2962   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
2963   <saml:Issuer>https://www.opensaml.org/IDP</saml:Issuer>
2964   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2965     <ds:SignedInfo>
2966       <ds:CanonicalizationMethod
2967         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
2968       <ds:SignatureMethod
2969         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
2970       <ds:Reference URI="#_c7055387-af61-4fce-8b98-e2927324b306">
2971         <ds:Transforms>
2972           <ds:Transform
2973             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
2974 signature" />
2975           <ds:Transform
2976             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
2977             <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
2978               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
2979           </ds:Transform>
2980         </ds:Transforms>
2981         <ds:DigestMethod
2982           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
2983         <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
2984       </ds:Reference>
2985     </ds:SignedInfo>
2986     <ds:SignatureValue>
2987       x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
2988       EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
2989       w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXBjJwqv5pUaE4=
2990     </ds:SignatureValue>
2991     <ds:KeyInfo>
2992       <ds:X509Data>
2993         <ds:X509Certificate>
2994           MIIICyJCCAjoGAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakkCzAJBgNVBAYTA1VT
2995           MRIwEAYDVQQIEWlXaXNjb25zaW4xEDAOBgNVBAcTB01hZGlzb24xIDAeBgNVBAoT
2996           FlVuaXZlcnNpdHkgb2YgV2l2Y29uc2luMSswKQYDVQQLEyJEaXZpc2l2biBvZiBJ
2997           bmZvcmlhdGl2biBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXXJ2ZXIgc0Eg
2998           LS0gMjAwMjA3MDFBMB4XDTAyMDcyNjA3Mjc1MVoXDTA2MDkwNDA3Mjc1MVowgYsx
2999           CzAJBgNVBAYTA1VTMREwDwYDVQQIEWhNaWNNoaWdhbjESMBAGA1UEBxMJQW5uIEFy
```

```

3000 Ym9yMQ4wDAYDVQKKEwVvQ0FJRDEcMBoGA1UEAxMTc2hpYjEuaW50ZXJ1ZmVkbWVkdTEuZWR1MIGfMA0G
3001 dTEuZWR1MIGfMA0GCSqGSIb3DQEJARYYcm9vdEBzaGlzMS5pbnRlcm5ldDIuZWZWR1MIGfMA0G
3002 CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDZSAb2sxvhaXnXVIVTx8vuRay+x50z7GJj
3003 IHRyQgIv6IqaGG04eTcyVMhoeKE0b45QgvBIAOAPSZB113R6+KYiE7x4XAWIrCP+
3004 c2MZVeXeTgV3Yz+USLg2Y1on+Jh4HxwkPFmZBctyXiUr6DxF8rvoP9W027rhRjE
3005 pmqOI fGTWQIDAQABox0wGzAMBgNVHRMBAf8EAjAAMAsGA1UdDwQEAwIFoDANBgkq
3006 hkiG9w0BAQQFAAOBgQBfDqEW+OI3jqBQHIBzhujN/PizdN7s/z4D5d3pptWDJf2n
3007 qgi7lFV6MDkHmTvTqBtjmNk3No7v/dnP6Hr7wHxvCCRwubnmIfz6QZAv2FU78pLX
3008 8I3bsbmRAUg4UP9hH6ABVq4KQKMknxulxQxLhpR1ylGPdiowMNTREg8cCx3w/w==
3009 </ds:X509Certificate>
3010 </ds:X509Data>
3011 </ds:KeyInfo>
3012 </ds:Signature>
3013 <Status>
3014   <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
3015 </Status>
3016 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
3017   IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
3018   xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
3019   <Issuer>https://www.opensaml.org/IDP</Issuer>
3020   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3021     <ds:SignedInfo>
3022       <ds:CanonicalizationMethod
3023         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3024       <ds:SignatureMethod
3025         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
3026       <ds:Reference URI="#_a75adf55-01d7-40cc-929f-dbd8372ebdfc">
3027         <ds:Transforms>
3028           <ds:Transform
3029             Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
3030 signature"/>
3031           <ds:Transform
3032             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
3033             <InclusiveNamespaces
3034               PrefixList="#default saml ds xs xsi"
3035               xmlns="http://www.w3.org/2001/10/xml-exc-c14n#">
3036             </ds:Transform>
3037           </ds:Transforms>
3038           <ds:DigestMethod
3039             Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
3040           <ds:DigestValue>Kclet6XcaOgOWXM4gty6/UNdviI=</ds:DigestValue>
3041         </ds:Reference>
3042       </ds:SignedInfo>
3043       <ds:SignatureValue>
3044         hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
3045         7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtpt3TD
3046         MwUL/cBUj2OtBZQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
3047       </ds:SignatureValue>
3048     </ds:Signature>
3049     <ds:KeyInfo>
3050       <ds:X509Data>
3051         <ds:X509Certificate>
3052           MIIICyJCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgaksCzAJBgNVBAYTA1VT
3053           MRIwEAYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24xIDAeBgNVBAoT
3054           FlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3055           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3056           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3057           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3058           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3059           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3060           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3061           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3062           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3063           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3064           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3065           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3066           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3067           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3068           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3069           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3070           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3071           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3072           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3073           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3074           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3075           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3076           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3077           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3078           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3079           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3080           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3081           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3082           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3083           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3084           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3085           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3086           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3087           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3088           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3089           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3090           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3091           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3092           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3093           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3094           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3095           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3096           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3097           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3098           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3099           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3100           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3101           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3102           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3103           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3104           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3105           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3106           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3107           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3108           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3109           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3110           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3111           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3112           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3113           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3114           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3115           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3116           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3117           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3118           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3119           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3120           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3121           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3122           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3123           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3124           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3125           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3126           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3127           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3128           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3129           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3130           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3131           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3132           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3133           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3134           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3135           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3136           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3137           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3138           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3139           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3140           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3141           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3142           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3143           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3144           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3145           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3146           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3147           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3148           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3149           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3150           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3151           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3152           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3153           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3154           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3155           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3156           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3157           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3158           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3159           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3160           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3161           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3162           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3163           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3164           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3165           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3166           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3167           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3168           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3169           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3170           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3171           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3172           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3173           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3174           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3175           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3176           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3177           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3178           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3179           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3180           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3181           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3182           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3183           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3184           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3185           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3186           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3187           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3188           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3189           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3190           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3191           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3192           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3193           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3194           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3195           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3196           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3197           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3198           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3199           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3200           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3201           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3202           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3203           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3204           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3205           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3206           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3207           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3208           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3209           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3210           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3211           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3212           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3213           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3214           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3215           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3216           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3217           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3218           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3219           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3220           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3221           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3222           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3223           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3224           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3225           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3226           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3227           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3228           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3229           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3230           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3231           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3232           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3233           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3234           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3235           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3236           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3237           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3238           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3239           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3240           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3241           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3242           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3243           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3244           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3245           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3246           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21zY29uc2luMSswKQYDVQQLIEYjEaXZpc21vbiBvZiBJ
3247           bmZvcmlhdGlvbiBUZWNobm9sb2d5MSUwIwYDVQQDExxIRVBLSSBTZXJ2ZXIgc0Eg
3248           LS0gMjAwMjA3MDZBMB4XDTEyMDcyNjA3Mjc1MVoXDTA2MDkxNDkxMjc1MVoYYSYx
3249           CzAJBgNVBAYTA1VTREwDwYDVQQIEw1XaXNjb25zaW4xEDAOBgNVBAcTB01hZG1zb24x
3250           IDAeBgNVBAoTFlVuaXZlcnNpdHkgb2YgV21z
```

```
3066         </ds:X509Certificate>
3067     </ds:X509Data>
3068     </ds:KeyInfo>
3069 </ds:Signature>
3070 <Subject>
3071     <NameID
3072         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
3073         scott@example.org
3074     </NameID>
3075     <SubjectConfirmation
3076         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
3077 </Subject>
3078 <Conditions NotBefore="2003-04-17T00:46:02Z"
3079     NotOnOrAfter="2003-04-17T00:51:02Z">
3080     <AudienceRestriction>
3081         <Audience>http://www.opensaml.org/SP</Audience>
3082     </AudienceRestriction>
3083 </Conditions>
3084 <AuthnStatement AuthnInstant="2003-04-17T00:46:00Z">
3085     <AuthnContext>
3086         <AuthnContextClassRef>
3087             urn:oasis:names:tc:SAML:2.0:ac:classes:Password
3088         </AuthnContextClassRef>
3089     </AuthnContext>
3090 </AuthnStatement>
3091 </Assertion>
3092 </Response>
```

## 6 SAML and XML Encryption Syntax and Processing

Encryption is used as the means to implement confidentiality. The most common motives for confidentiality are to protect the personal privacy of individuals or to protect organizational secrets for competitive advantage or similar reasons. Confidentiality may also be required to ensure the effectiveness of some other security mechanism. For example, a secret password or key may be encrypted.

Several ways of using encryption to confidentially protect all or part of a SAML assertion are provided.

- Communications confidentiality may be provided by mechanisms associated with a particular binding or profile. For example, the SOAP Binding [SAMLBind] supports the use of SSL/TLS (see [RFC 2246]/[SSL3]) or SOAP Message Security mechanisms for confidentiality.
- A `<SubjectConfirmation>` secret can be protected through the use of the `<ds:KeyInfo>` element within `<SubjectConfirmationData>`, which permits keys or other secrets to be encrypted.
- An entire `<Assertion>` element may be encrypted, as described in Section 2.3.4.
- The `<BaseID>` or `<NameID>` element may be encrypted, as described in Section 2.2.4.
- An `<Attribute>` element may be encrypted, as described in Section 2.7.3.2.

### 6.1 General Considerations

Encryption of the `<Assertion>`, `<BaseID>`, `<NameID>` and `<Attribute>` elements is provided by use of XML Encryption [XMLEnc]. Encrypted data and optionally one or more encrypted keys MUST replace the plaintext information in the same location within the XML instance. The `<EncryptedData>` element's `Type` attribute SHOULD be used and, if it is present, MUST have the value `http://www.w3.org/2001/04/xmlenc#Element`.

Any of the algorithms defined for use with XML Encryption MAY be used to perform the encryption. The SAML schema is defined so that the inclusion of the encrypted data yields a valid instance.

### 6.2 Combining Signatures and Encryption

Use of XML Encryption and XML Signature MAY be combined. When an assertion is to be signed and encrypted, the following rules apply. A relying party MUST perform signature validation and decryption in the reverse order that signing and encryption were performed.

- When a signed `<Assertion>` element is encrypted, the signature MUST first be calculated and placed within the `<Assertion>` element before the element is encrypted.
- When a `<BaseID>`, `<NameID>`, or `<Attribute>` element is encrypted, the encryption MUST be performed first and then the signature calculated over the assertion or message containing the encrypted element.

## 7 SAML Extensibility

SAML supports extensibility in a number of ways, including extending the assertion and protocol schemas. An example of an application that extends SAML assertions is the Liberty Protocols and Schema Specification [LibertyProt]. The following sections explain the extensibility features with SAML assertions and protocols.

See the SAML Profiles specification [SAMLProf] for information on how to define new profiles, which can be combined with extensions to put the SAML framework to new uses.

### 7.1 Schema Extension

Note that elements in the SAML schemas are blocked from substitution, which means that no SAML elements can serve as the head element of a substitution group. However, SAML types are not defined as *final*, so that all SAML types MAY be extended and restricted. As a practical matter, this means that extensions are typically defined only as types rather than elements, and are included in SAML instances by means of an `xsi:type` attribute.

The following sections discuss only elements and types that have been specifically designed to support extensibility.

#### 7.1.1 Assertion Schema Extension

The SAML assertion schema (see [SAML-XSD]) is designed to permit separate processing of the assertion package and the statements it contains, if the extension mechanism is used for either part.

The following elements are intended specifically for use as extension points in an extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived type:

- `<BaseID>` and **BaseIDAbstractType**
- `<Condition>` and **ConditionAbstractType**
- `<Statement>` and **StatementAbstractType**
- The following constructs that are directly usable as part of SAML are particularly interesting targets for extension:
  - `<AuthnStatement>` and **AuthnStatementType**
  - `<AttributeStatement>` and **AttributeStatementType**
  - `<AuthzDecisionStatement>` and **AuthzDecisionStatementType**
  - `<AudienceRestriction>` and **AudienceRestrictionType**
  - `<ProxyRestriction>` and **ProxyRestrictionType**
  - `<OneTimeUse>` and **OneTimeUseType**

#### 7.1.2 Protocol Schema Extension

The following SAML protocol elements are intended specifically for use as extension points in an extension schema; their types are set to *abstract*, and are thus usable only as the base of a derived type:

- `<Request>` and **RequestAbstractType**
- `<SubjectQuery>` and **SubjectQueryAbstractType**



3162 The following constructs that are directly usable as part of SAML are particularly interesting targets for  
 3163 extension:

- 3164 • `<AuthnQuery>` and **AuthnQueryType**
- 3165 • `<AuthzDecisionQuery>` and **AuthzDecisionQueryType**
- 3166 • `<AttributeQuery>` and **AttributeQueryType**
- 3167 • **StatusResponseType**

## 3168 7.2 Schema Wildcard Extension Points

3169 The SAML schemas use wildcard constructs in some locations to allow the use of elements and attributes  
 3170 from arbitrary namespaces, which serves as a built-in extension point without requiring an extension  
 3171 schema.

### 3172 7.2.1 Assertion Extension Points

3173 The following constructs in the assertion schema allow constructs from arbitrary namespaces within them:

- 3174 • `<SubjectConfirmationData>`: Uses **xs:anyType**, which allows any sub-elements and  
 3175 attributes.
- 3176 • `<AuthnContextDecl>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3177 • `<AttributeValue>`: Uses **xs:anyType**, which allows any sub-elements and attributes.
- 3178 • `<Advice>` and **AdviceType**: In addition to SAML-native elements, allows elements from other  
 3179 namespaces with lax schema validation processing.

3180 The following constructs in the assertion schema allow arbitrary global attributes:

- 3181 • `<Attribute>` and **AttributeType**

### 3182 7.2.2 Protocol Extension Points

3183 The following constructs in the protocol schema allow constructs from arbitrary namespaces within them:

- 3184 • `<Extensions>` and **ExtensionsType**: Allows elements from other namespaces with lax schema  
 3185 validation processing.
- 3186 • `<StatusDetail>` and **StatusDetailType**: Allows elements from other namespaces with lax  
 3187 schema validation processing.
- 3188 • `<ArtifactResponse>` and **ArtifactResponseType**: Allows elements from any namespaces with  
 3189 lax schema validation processing. (It is specifically intended to carry a SAML request or response  
 3190 message element, however.)

## 3191 7.3 Identifier Extension

3192 SAML uses URI-based identifiers for a number of purposes, such as status codes and name identifier  
 3193 formats, and defines some identifiers that MAY be used for these purposes; most are listed in Section 8.  
 3194 However, it is always possible to define additional URI-based identifiers for these purposes. It is  
 3195 RECOMMENDED that these additional identifiers be defined in a formal profile of use. In no case should  
 3196 the meaning of a given URI used as such an identifier significantly change, or be used to mean two  
 3197 different things.

## 8 SAML-Defined Identifiers

The following sections define URI-based identifiers for common resource access actions, subject name identifier formats, and attribute name formats.

Where possible an existing URN is used to specify a protocol. In the case of IETF protocols, the URN of the most current RFC that specifies the protocol is used. URI references created specifically for SAML have one of the following stems, according to the specification set version in which they were first introduced:

```
urn:oasis:names:tc:SAML:1.0:
urn:oasis:names:tc:SAML:1.1:
urn:oasis:names:tc:SAML:2.0:
```

### 8.1 Action Namespace Identifiers

The following identifiers MAY be used in the `Namespace` attribute of the `<Action>` element to refer to common sets of actions to perform on resources.

#### 8.1.1 Read/Write/Execute/Delete/Control

**URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc`

Defined actions:

`Read Write Execute Delete Control`

These actions are interpreted as follows:

**Read**

The subject may read the resource.

**Write**

The subject may modify the resource.

**Execute**

The subject may execute the resource.

**Delete**

The subject may delete the resource.

**Control**

The subject may specify the access control policy for the resource.

#### 8.1.2 Read/Write/Execute/Delete/Control with Negation

**URI:** `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`

Defined actions:

`Read Write Execute Delete Control ~Read ~Write ~Execute ~Delete ~Control`

The actions specified in Section 8.1.1 are interpreted in the same manner described there. Actions prefixed with a tilde (~) are negated permissions and are used to affirmatively specify that the stated permission is denied. Thus a subject described as being authorized to perform the action `~Read` is affirmatively denied read permission.

3234 A SAML authority MUST NOT authorize both an action and its negated form.

### 3235 8.1.3 Get/Head/Put/Post

3236 **URI:** urn:oasis:names:tc:SAML:1.0:action:ghpp

3237 Defined actions:

3238 GET HEAD PUT POST

3239 These actions bind to the corresponding HTTP operations. For example a subject authorized to perform  
3240 the GET action on a resource is authorized to retrieve it.

3241 The GET and HEAD actions loosely correspond to the conventional read permission and the PUT and POST  
3242 actions to the write permission. The correspondence is not exact however since an HTTP GET operation  
3243 may cause data to be modified and a POST operation may cause modification to a resource other than  
3244 the one specified in the request. For this reason a separate Action URI reference specifier is provided.

### 3245 8.1.4 UNIX File Permissions

3246 **URI:** urn:oasis:names:tc:SAML:1.0:action:unix

3247 The defined actions are the set of UNIX file access permissions expressed in the numeric (octal) notation.

3248 The action string is a four-digit numeric code:

3249 *extended user group world*

3250 Where the *extended* access permission has the value

3251 +2 if sgid is set

3252 +4 if suid is set

3253 The *user group* and *world* access permissions have the value

3254 +1 if execute permission is granted

3255 +2 if write permission is granted

3256 +4 if read permission is granted

3257 For example, 0754 denotes the UNIX file access permission: user read, write, and execute; group read  
3258 and execute; and world read.

## 3259 8.2 Attribute Name Format Identifiers

3260 The following identifiers MAY be used in the NameFormat attribute defined on the **AttributeType** complex  
3261 type to refer to the classification of the attribute name for purposes of interpreting the name.

### 3262 8.2.1 Unspecified

3263 **URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified

3264 The interpretation of the attribute name is left to individual implementations.

## 8.2.2 URI Reference

**URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:uri

The attribute name follows the convention for URI references [RFC 2396], for example as used in XACML [XACML] attribute identifiers. The interpretation of the URI content or naming scheme is application-specific. See [SAMLProf] for attribute profiles that make use of this identifier.

## 8.2.3 Basic

**URI:** urn:oasis:names:tc:SAML:2.0:attrname-format:basic

The class of strings acceptable as the attribute name **MUST** be drawn from the set of values belonging to the primitive type **xs:Name** as defined in [Schema2] Section 3.3.6. See [SAMLProf] for attribute profiles that make use of this identifier.

## 8.3 Name Identifier Format Identifiers

The following identifiers **MAY** be used in the `Format` attribute of the `<NameID>`, `<NameIDPolicy>`, or `<Issuer>` elements (see Section 2.2) to refer to common formats for the content of the elements and the associated processing rules, if any.

**Note:** Several identifiers that were deprecated in SAML V1.1 have been removed for SAML V2.0.

### 8.3.1 Unspecified

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

The interpretation of the content of the element is left to individual implementations.

### 8.3.2 Email Address

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Indicates that the content of the element is in the form of an email address, specifically "addr-spec" as defined in IETF RFC 2822 [RFC 2822] Section 3.4.1. An addr-spec has the form local-part@domain. Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">".

### 8.3.3 X.509 Subject Name

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

Indicates that the content of the element is in the form specified for the contents of the `<ds:X509SubjectName>` element in the XML Signature Recommendation [XMLSig]. Implementors should note that the XML Signature specification specifies encoding rules for X.509 subject names that differ from the rules given in IETF RFC 2253 [RFC 2253].

### 8.3.4 Windows Domain Qualified Name

**URI:** urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

3298 Indicates that the content of the element is a Windows domain qualified name. A Windows domain  
 3299 qualified user name is a string of the form "DomainName\UserName". The domain name and "\" separator  
 3300 MAY be omitted.

### 3301 **8.3.5 Kerberos Principal Name**

3302 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

3303 Indicates that the content of the element is in the form of a Kerberos principal name using the format  
 3304 name[/instance]@REALM. The syntax, format and characters allowed for the name, instance, and  
 3305 realm are described in IETF RFC 1510 [RFC 1510].

### 3306 **8.3.6 Entity Identifier**

3307 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:entity

3308 Indicates that the content of the element is the identifier of an entity that provides SAML-based services  
 3309 (such as a SAML authority, requester, or responder) or is a participant in SAML profiles (such as a service  
 3310 provider supporting the browser SSO profile). Such an identifier can be used in the <Issuer> element to  
 3311 identify the issuer of a SAML request, response, or assertion, or within the <NameID> element to make  
 3312 assertions about system entities that can issue SAML requests, responses, and assertions. It can also be  
 3313 used in other elements and attributes whose purpose is to identify a system entity in various protocol  
 3314 exchanges.

3315 The syntax of such an identifier is a URI of not more than 1024 characters in length. It is  
 3316 RECOMMENDED that a system entity use a URL containing its own domain name to identify itself.

3317 The `NameQualifier`, `SPNameQualifier`, and `SPProvidedID` attributes MUST be omitted.

### 3318 **8.3.7 Persistent Identifier**

3319 **URI:** urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

3320 Indicates that the content of the element is a persistent opaque identifier for a principal that is specific to  
 3321 an identity provider and a service provider or affiliation of service providers. Persistent name identifiers  
 3322 generated by identity providers MUST be constructed using pseudo-random values that have no  
 3323 discernible correspondence with the subject's actual identifier (for example, username). The intent is to  
 3324 create a non-public, pair-wise pseudonym to prevent the discovery of the subject's identity or activities.  
 3325 Persistent name identifier values MUST NOT exceed a length of 256 characters.

3326 The element's `NameQualifier` attribute, if present, MUST contain the unique identifier of the identity  
 3327 provider that generated the identifier (see Section 8.3.6). It MAY be omitted if the value can be derived  
 3328 from the context of the message containing the element, such as the issuer of a protocol message or an  
 3329 assertion containing the identifier in its subject. Note that a different system entity might later issue its own  
 3330 protocol message or assertion containing the identifier; the `NameQualifier` attribute does not change in  
 3331 this case, but MUST continue to identify the entity that originally created the identifier (and MUST NOT be  
 3332 omitted in such a case).

3333 The element's `SPNameQualifier` attribute, if present, MUST contain the unique identifier of the service  
 3334 provider or affiliation of providers for whom the identifier was generated (see Section 8.3.6). It MAY be  
 3335 omitted if the element is contained in a message intended only for consumption directly by the service  
 3336 provider, and the value would be the unique identifier of that service provider.

3337 The element's `SPProvidedID` attribute MUST contain the alternative identifier of the principal most  
 3338 recently set by the service provider or affiliation, if any (see Section 3.6). If no such identifier has been  
 3339 established, then the attribute MUST be omitted.

Persistent identifiers are intended as a privacy protection mechanism; as such they MUST NOT be shared in clear text with providers other than the providers that have established the shared identifier. Furthermore, they MUST NOT appear in log files or similar locations without appropriate controls and protections. Deployments without such requirements are free to use other kinds of identifiers in their SAML exchanges, but MUST NOT overload this format with persistent but non-opaque values

Note also that while persistent identifiers are typically used to reflect an account linking relationship between a pair of providers, a service provider is not obligated to recognize or make use of the long term nature of the persistent identifier or establish such a link. Such a "one-sided" relationship is not discernibly different and does not affect the behavior of the identity provider or any processing rules specific to persistent identifiers in the protocols defined in this specification.

Finally, note that the `NameQualifier` and `SPNameQualifier` attributes indicate directionality of creation, but not of use. If a persistent identifier is created by a particular identity provider, the `NameQualifier` attribute value is permanently established at that time. If a service provider that receives such an identifier takes on the role of an identity provider and issues its own assertion containing that identifier, the `NameQualifier` attribute value does not change (and would of course not be omitted). It might alternatively choose to create its own persistent identifier to represent the principal and link the two values. This is a deployment decision.

### 8.3.8 Transient Identifier

**URI:** `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`

Indicates that the content of the element is an identifier with transient semantics and SHOULD be treated as an opaque and temporary value by the relying party. Transient identifier values MUST be generated in accordance with the rules for SAML identifiers (see Section 1.3.4), and MUST NOT exceed a length of 256 characters.

The `NameQualifier` and `SPNameQualifier` attributes MAY be used to signify that the identifier represents a transient and temporary pair-wise identifier. In such a case, they MAY be omitted in accordance with the rules specified in Section 8.3.7.

## 8.4 Consent Identifiers

The following identifiers MAY be used in the `Consent` attribute defined on the **RequestAbstractType** and **StatusResponseType** complex types to communicate whether a principal gave consent, and under what conditions, for the message.

### 8.4.1 Unspecified

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:unspecified`

No claim as to principal consent is being made.

### 8.4.2 Obtained

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:obtained`

Indicates that a principal's consent has been obtained by the issuer of the message.

### 8.4.3 Prior

**URI:** `urn:oasis:names:tc:SAML:2.0:consent:prior`



3378 Indicates that a principal's consent has been obtained by the issuer of the message at some point prior to  
3379 the action that initiated the message.

#### 3380 **8.4.4 Implicit**

3381 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-implicit

3382 Indicates that a principal's consent has been implicitly obtained by the issuer of the message during the  
3383 action that initiated the message, as part of a broader indication of consent. Implicit consent is typically  
3384 more proximal to the action in time and presentation than prior consent, such as part of a session of  
3385 activities.

#### 3386 **8.4.5 Explicit**

3387 **URI:** urn:oasis:names:tc:SAML:2.0:consent:current-explicit

3388 Indicates that a principal's consent has been explicitly obtained by the issuer of the message during the  
3389 action that initiated the message.

#### 3390 **8.4.6 Unavailable**

3391 **URI:** urn:oasis:names:tc:SAML:2.0:consent:unavailable

3392 Indicates that the issuer of the message did not obtain consent.

#### 3393 **8.4.7 Inapplicable**

3394 **URI:** urn:oasis:names:tc:SAML:2.0:consent:inapplicable

3395 Indicates that the issuer of the message does not believe that they need to obtain or report consent.

## 9 References

The following works are cited in the body of this specification.

### 9.1 Normative References

- [Excl-C14N]** J. Boyer et al. *Exclusive XML Canonicalization Version 1.0*. World Wide Web Consortium, July 2002. See <http://www.w3.org/TR/xml-exc-c14n/>.
- [Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-1/>. Note that this specification normatively references [Schema2], listed below.
- [Schema2]** P. V. Biron et al. *XML Schema Part 2: Datatypes*. World Wide Web Consortium Recommendation, May 2001. See <http://www.w3.org/TR/xmlschema-2/>.
- [XML]** T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium, October 2000. See <http://www.w3.org/TR/REC-xml>.
- [XMLEnc]** D. Eastlake et al. *XML Encryption Syntax and Processing*. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>. Note that this specification normatively references [XMLEnc-XSD], listed below.
- [XMLEnc-XSD]** XML Encryption Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>.
- [XMLNS]** T. Bray et al. *Namespaces in XML*. World Wide Web Consortium, January 1999. See <http://www.w3.org/TR/REC-xml-names>.
- [XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web Consortium, February 2002. See <http://www.w3.org/TR/xmldsig-core/>. Note that this specification normatively references [XMLSig-XSD], listed below.
- [XMLSig-XSD]** XML Signature Schema. World Wide Web Consortium. See <http://www.w3.org/TR/2000/CR-xmldsig-core-20001031/xmldsig-core-schema.xsd>.

### 9.2 Non-Normative References

- [LibertyProt]** J. Beatty et al. *Liberty Protocols and Schema Specification Version 1.1*. Liberty Alliance Project, January 2003. See [http://www.projectliberty.org/specs/archive/v1\\_1/liberty-architecture-protocols-schema-v1.1.pdf](http://www.projectliberty.org/specs/archive/v1_1/liberty-architecture-protocols-schema-v1.1.pdf).
- [RFC 1510]** J. Kohl, C. Neuman. *The Kerberos Network Authentication Requestor (V5)*. IETF RFC 1510, September 1993. See <http://www.ietf.org/rfc/rfc1510.txt>.
- [RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- [RFC 2246]** T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See <http://www.ietf.org/rfc/rfc2246.txt>.
- [RFC 2253]** M. Wahl et al. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*. IETF RFC 2253, December 1997. See <http://www.ietf.org/rfc/rfc2253.txt>.
- [RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396, August, 1998. See <http://www.ietf.org/rfc/rfc2396.txt>.
- [RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. See <http://www.ietf.org/rfc/rfc2822.txt>.

3439	<b>[RFC 3075]</b>	D. Eastlake, J. Reagle, D. Solo. <i>XML-Signature Syntax and Processing</i> . IETF RFC 3075, March 2001. See <a href="http://www.ietf.org/rfc/rfc3075.txt">http://www.ietf.org/rfc/rfc3075.txt</a> .
3440		
3441	<b>[RFC 3513]</b>	R. Hinden, S. Deering, <i>Internet Protocol Version 6 (IPv6) Addressing Architecture</i> . IETF RFC 3513, April 2003. See <a href="http://www.ietf.org/rfc/rfc3513.txt">http://www.ietf.org/rfc/rfc3513.txt</a> .
3442		
3443	<b>[SAMLAuthnCxt]</b>	J. Kemp et al. <i>Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-authn-context-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3444		
3445		
3446	<b>[SAMLBind]</b>	S. Cantor et al. <i>Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3447		
3448		
3449	<b>[SAMLConform]</b>	P. Mishra et al. <i>Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3450		
3451		
3452	<b>[SAMLGloss]</b>	J. Hodges et al. <i>Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3453		
3454		
3455	<b>[SAMLMeta]</b>	S. Cantor et al. <i>Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3456		
3457		
3458	<b>[SAMLXSD]</b>	S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3459		
3460		
3461	<b>[SAMLProf]</b>	S. Cantor et al. <i>Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-profiles-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3462		
3463		
3464	<b>[SAMLSecure]</b>	F. Hirsch et al. <i>Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0</i> . OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3465		
3466		
3467		
3468	<b>[SAMLTechOvw]</b>	J. Hughes et al. SAML Technical Overview. OASIS, February 2005. Document ID sstc-saml-tech-overview-2.0-draft-03. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3469		
3470		
3471	<b>[SAMLXSD]</b>	S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See <a href="http://www.oasis-open.org/committees/security/">http://www.oasis-open.org/committees/security/</a> .
3472		
3473		
3474	<b>[SSL3]</b>	A. Frier et al. <i>The SSL 3.0 Protocol</i> . Netscape Communications Corp, November 1996.
3475		
3476	<b>[UNICODE-C]</b>	M. Davis, M. J. Dürst. <i>Unicode Normalization Forms</i> . UNICODE Consortium, March 2001. See <a href="http://www.unicode.org/unicode/reports/tr15/tr15-21.html">http://www.unicode.org/unicode/reports/tr15/tr15-21.html</a> .
3477		
3478	<b>[W3C-CHAR]</b>	M. J. Dürst. <i>Requirements for String Identity Matching and String Indexing</i> . World Wide Web Consortium, July 1998. See <a href="http://www.w3.org/TR/WD-charreq">http://www.w3.org/TR/WD-charreq</a> .
3479		
3480	<b>[W3C-CharMod]</b>	M. J. Dürst. <i>Character Model for the World Wide Web 1.0: Normalization</i> . World Wide Web Consortium, February 2004. See <a href="http://www.w3.org/TR/charmod-norm/">http://www.w3.org/TR/charmod-norm/</a> .
3481		
3482		
3483	<b>[XACML]</b>	eXtensible Access Control Markup Language (XACML), product of the OASIS XACML TC. See <a href="http://www.oasis-open.org/committees/xacml">http://www.oasis-open.org/committees/xacml</a> .
3484		
3485	<b>[XML-ID]</b>	J. Marsh et al. <i>xml:id Version 1.0</i> , World Wide Web Consortium, April 2004. See <a href="http://www.w3.org/TR/xml-id/">http://www.w3.org/TR/xml-id/</a> .
3486		

## Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services Technical Committee, whose voting members at the time of publication were:

- Conor Cahill, AOL
- John Hughes, Atos Origin
- Hal Lockhart, BEA Systems
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Gavenraj Sodhi, Computer Associates
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiestler, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems

- 3529 • Eve Maler, Sun Microsystems
- 3530 • Ronald Monzillo, Sun Microsystems
- 3531 • Emily Xu, Sun Microsystems
- 3532 • Greg Whitehead, Trustgenix

3533

3534 The editors also would like to acknowledge the following former SSTC members for their contributions to  
 3535 this or previous versions of the OASIS Security Assertions Markup Language Standard:

- 3536 • Stephen Farrell, Baltimore Technologies
- 3537 • David Orchard, BEA Systems
- 3538 • Krishna Sankar, Cisco Systems
- 3539 • Zahid Ahmed, CommerceOne
- 3540 • Tim Alsop, CyberSafe Limited
- 3541 • Carlisle Adams, Entrust
- 3542 • Tim Moses, Entrust
- 3543 • Nigel Edwards, Hewlett-Packard
- 3544 • Joe Pato, Hewlett-Packard
- 3545 • Bob Blakley, IBM
- 3546 • Marlena Erdos, IBM
- 3547 • Marc Chanliau, Netegrity
- 3548 • Chris McLaren, Netegrity
- 3549 • Lynne Rosenthal, NIST
- 3550 • Mark Skall, NIST
- 3551 • Charles Knouse, Oblix
- 3552 • Simon Godik, Overxeer
- 3553 • Charles Norwood, SAIC
- 3554 • Evan Prodromou, Securant
- 3555 • Robert Griffin, RSA Security (former editor)
- 3556 • Sai Allarvarpu, Sun Microsystems
- 3557 • Gary Ellison, Sun Microsystems
- 3558 • Chris Ferris, Sun Microsystems
- 3559 • Mike Myers, Traceroute Security
- 3560 • Phillip Hallam-Baker, VeriSign (former editor)
- 3561 • James Vanderbeek, Vodafone
- 3562 • Mark O'Neill, Vordel
- 3563 • Tony Palmer, Vordel

3564

3565 Finally, the editors wish to acknowledge the following people for their contributions of material used as  
 3566 input to the OASIS Security Assertions Markup Language specifications:

- 3567 • Thomas Gross, IBM
- 3568 • Birgit Pfitzmann, IBM

## Appendix B. Notices

3569

3570 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
 3571 might be claimed to pertain to the implementation or use of the technology described in this document or  
 3572 the extent to which any license under such rights might or might not be available; neither does it represent  
 3573 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to  
 3574 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made  
 3575 available for publication and any assurances of licenses to be made available, or the result of an attempt  
 3576 made to obtain a general license or permission for the use of such proprietary rights by implementors or  
 3577 users of this specification, can be obtained from the OASIS Executive Director.

3578 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or  
 3579 other proprietary rights which may cover technology that may be required to implement this specification.  
 3580 Please address the information to the OASIS Executive Director.

3581 **Copyright © OASIS Open 2005. All Rights Reserved.**

3582 This document and translations of it may be copied and furnished to others, and derivative works that  
 3583 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and  
 3584 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and  
 3585 this paragraph are included on all such copies and derivative works. However, this document itself may  
 3586 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as  
 3587 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights  
 3588 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it  
 3589 into languages other than English.

3590 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
 3591 or assigns.

3592 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
 3593 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
 3594 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
 3595 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.



# EXHIBIT 11

# **APPLIED CRYPTOGRAPHY, SECOND EDITION**

**PROTOCOLS, ALGORITHMS, AND SOURCE CODE IN C**

**BRUCE SCHNEIER**



John Wiley & Sons, Inc.

New York • Chichester • Brisbane • Toronto • Singapore



QA76  
.9  
.A25S35  
1996  
copy 2

Publisher: Katherine Schowalter  
Editor: Phil Sutherland  
Assistant Editor: Allison Roarty  
Managing Editor: Robert Aronds  
Text Design & Composition: North Market Street Graphics

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc. is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

This text is printed on acid-free paper.

Copyright © 1996 by Bruce Schneier  
Published by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional service. If legal advice or other expert assistance is required, the services of a competent professional person should be sought.

In no event will the publisher or author be liable for any consequential, incidental, or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising from the use or inability to use the protocols and algorithms in this book, even if the publisher or author has been advised of the possibility of such damages.

Some of the protocols and algorithms in this book are protected by patents and copyrights. It is the responsibility of the reader to obtain all necessary patent and copyright licenses before implementing in software any protocol or algorithm in this book. This book does not contain an exhaustive list of all applicable patents and copyrights.

Some of the protocols and algorithms in this book are regulated under the United States Department of State International Traffic in Arms Regulations. It is the responsibility of the reader to obtain all necessary export licenses before implementing in software for export any protocol or algorithm in this book.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

***Library of Congress Cataloging-in-Publication Data:***

Schneier, Bruce  
Applied Cryptography Second Edition : protocols, algorithms, and source code in C  
/ Bruce Schneier.  
p. cm.  
Includes bibliographical references (p. 675).  
ISBN 0-471-12845-7 (cloth : acid-free paper). — ISBN  
0-471-11709-9 (paper : acid-free paper)  
1. Computer security. 2. Telecommunication—Security measures.  
3. Cryptography. I. Title.  
QA76.9.A25S35 1996  
005.8'2—dc20

95-12398  
CIP

Printed in the United States of America  
10 9 8 7 6 5 4 3 2 1

## 24.14 PUBLIC-KEY CRYPTOGRAPHY STANDARDS (PKCS)

The Public-Key Cryptography Standards (PKCS) are RSA Data Security, Inc.'s attempt to provide an industry standard interface for public-key cryptography. Traditionally, this sort of thing would be handled by ANSI, but, considering the current situation in cryptography politics, RSADSI figured that they had better do it on their own. Working with a variety of companies, they developed a series of standards. Some are compatible with other standards and some are not.

These are not standards in the traditional sense of the word; no standards body convened and voted on PKCS. According to its own materials, RSADSI will "retain sole decision-making authority on what each standard is" and will "publish revised standards when appropriate" [803].

Even so, there is a lot of good stuff here. If you're not sure what kind of syntax and data structures to use when programming public-key cryptography, these standards are probably as good as anything else you can come up with. And, since they're not really standards, you can tailor them to suit your needs.

Following is a short description of each PKCS (PKCS #2 and PKCS #4 have been incorporated into PKCS #1).

PKCS #1 [1345] describes a method for RSA encryption and decryption, primarily for constructing the digital signatures and digital envelopes described in PKCS #7. For digital signatures, the message is hashed and then the hash is encrypted with the private key of the signer. Both message and hash are represented together as detailed in PKCS #7. For digital envelopes (encrypted messages), the message is first encrypted with a symmetric algorithm, and then the message key is encrypted with the public key of the recipient. The encrypted message and encrypted key are represented together according to the syntax of PKCS #7. Both of these methods are compatible with PEM standards. PKCS #1 also describes a syntax, identical to the syntax in X.509 and PEM, for RSA public and private keys and three signature algorithms—MD2 and RSA, MD4 and RSA, and MD5 and RSA—for signing certificates and the like.

PKCS #3 [1346] describes a method for implementing Diffie-Hellman key exchange.

PKCS #5 [1347] describes a method for encrypting messages with a secret key derived from a password. It uses either MD2 or MD5 to derive the key from the password, and encrypts with DES in CBC mode. The method is intended primarily to encrypt private keys when transferring them from one computer system to another, but can be used to encrypt messages.

PKCS #6 [1348] describes a standard syntax for public key certificates. The syntax is a superset of an X.509 certificate, so that X.509 certificates can be extracted if necessary. Over and above the X.509 set, additional attributes extend the certification process beyond just the public key. These include other information, such as electronic mail address.

PKCS #7 [1349] is a general syntax for data that may be encrypted or signed, such as digital envelopes or digital signatures. The syntax is recursive, so that envelopes can be nested, or someone can sign some previously encrypted data. The syntax also



allows other attributes, such as timestamps, to be authenticated along with the message content. PKCS #7 is compatible with PEM so that signed and encrypted messages can be converted to PEM messages without any cryptographic operations, and vice versa. PKCS #7 can support a variety of architectures—PEM is one—for certificate-based key management.

PKCS #8 [1350] describes a syntax for private key information—including a private key and a set of attributes—and a syntax for encrypted private keys. PKCS #5 can be used to encrypt the private key information.

PKCS #9 [1351] defines selected attribute types for PKCS #6 extended certificates, PKCS #7 digitally signed messages, and PKCS #8 private-key information.

PKCS #10 [1352] describes a standard syntax for certification requests. A certification comprises a distinguished name, a public key, and (optionally) a set of attributes, collectively signed by the person requesting certification. Certification requests are sent to a certification authority, who either transforms the request into an X.509 public-key certificate or a PKCS #6 certificate.

PKCS #11 [1353], the Cryptographic Token API Standard, specifies a programming interface called "Cryptoki" for portable cryptographic devices of all kinds. Cryptoki presents a common logical model, enabling applications to perform cryptographic operations on portable devices without knowing details of the underlying technology. The standard also defines application profiles: sets of algorithms that a device may support.

PKCS #12 [1354] describes syntax for storing in software a user's public keys, protected private keys, certificates, and other related cryptographic information. The goal is to standardize on a single key file for use among a variety of applications.

These standards are comprehensive, but not exhaustive. Many things are outside their scope: the problem of naming, noncryptographic issues surrounding certification, key lengths, and conditions on various parameters. What the PKCS provide are a format for transferring data based on public-key cryptography and an infrastructure to support that transfer.

## 24.15 UNIVERSAL ELECTRONIC PAYMENT SYSTEM (UEPS)

The UEPS is a smart-card banking application initially developed for rural South Africa, but later adopted by all of that country's major banking groups. About 2 million cards were issued in that country by early 1995. It has also been adopted in Namibia, and is also being deployed by at least one bank in Russia.

The system provides a secure debit card suitable for regions where poor telephone service make on-line verification impossible. Both customers and merchants have cards; customers can use their cards to transfer money to merchants. Merchants can then take their cards to a telephone and deposit the money in their bank account; customers can take their cards to a telephone and have money moved onto their card. There is no intention to provide anonymity, only to prevent fraud.

# EXHIBIT 12



UNITED STATES DISTRICT COURT  
SOUTHERN DISTRICT OF NEW YORK

WILLIAM GRECIA,

Plaintiff,

-v-

MASTERCARD INTERNATIONAL INC.,

Defendant.

No. 15-cv-9059 (RJS)  
ORDER

WILLIAM GRECIA,

Plaintiff,

-v-

SAMSUNG ELECTRONICS AMERICA,  
INC.,

Defendant.

No. 16-cv-9691 (RJS)  
ORDER

RICHARD J. SULLIVAN, District Judge:

Plaintiff William Grecia, the purported inventor of technologies used to prevent the unauthorized copying of digital media, such as movies, music, or games, brings these related actions against Defendants MasterCard International, Inc. (“MasterCard”) and Samsung Electronics America, Inc. (“Samsung”) for infringement of three of his patents: U.S. Patent Nos. 8,402,555 (“the ‘555 Patent”), 8,533,860 (“the ‘860 Patent”), and 8,887,308 (“the ‘308 Patent”).<sup>1</sup> Now before the Court are the parties’ competing constructions of Grecia’s patent claims pursuant to *Markman v. Westview Instruments, Inc.*, 517 U.S. 370 (1996). For the reasons set forth below, the Court adopts the following construction of the disputed patent terms.

<sup>1</sup> The patents share a common specification. Only the ‘860 Patent is asserted against Samsung.

## I. PROCEDURAL HISTORY<sup>2</sup>

On November 18, 2015, Grecia filed a complaint alleging that MasterCard infringes Claim 1 of the ‘360 Patent and various claims of the ‘555 and ‘860 Patents. (Doc. No. 1.) About a year later, on December 15, 2016, Grecia filed a complaint alleging that Samsung infringes various claims of the ‘860 Patent. (16-cv-9691, Doc. No. 1.) The Court accepted the action against Samsung as related to the earlier filed action against MasterCard.

In August 2017, the United States Patent and Trademark Office (“PTO”) granted MasterCard’s petition for *inter partes* review (“IPR”) of the ‘860 Patent, and instituted an IPR proceeding as to Claims 1–8 and 11–20 of that patent. (See Doc. No. 51.) In light of the pending IPR proceeding, MasterCard moved to stay this litigation. (*Id.*) Shortly thereafter, however, at Grecia’s request, the PTO entered an adverse judgment against Grecia on Claims 1–8 and 11–20 of the ‘860 Patent and terminated the IPR. (See Doc. No. 63.) Accordingly, the Court denied MasterCard’s motion to stay as moot. (Doc. No. 64.)

On September 25, 2017, the parties filed their Joint Claim Construction Chart. (Doc. No. 62.) Grecia filed his opening claim construction brief on October 23, 2017 (Doc. No. 65), Defendants filed their opposing brief on November 21, 2017 (Doc. No. 68), and Grecia filed his reply on November 28, 2017 (Doc. No. 69). Because Grecia’s reply exceeded the standard page limit by five pages, the Court granted Defendants leave to submit a five-page surreply (Doc. No. 72), which they filed on December 20, 2017 (Doc. No. 73). The Court held a claim construction hearing on May 24, 2018.

---

<sup>2</sup> In determining the proper construction of the patent claims, the Court has considered the parties’ Joint Claim Construction Chart, dated September 25, 2017 (Doc. No. 55), Grecia’s Opening Claim Construction brief (Doc. No. 65 (“Grecia Br.”), Defendants’ Claim Construction Response brief (Doc. No. 68 (“Def. Br.”)), Grecia’s Claim Construction Reply brief (Doc. No. 69 (“Grecia Reply”)), Defendants’ Claim Construction Surreply brief (Doc. No. 73 (“Def. Surreply”)), the materials attached to those submissions, and the parties’ presentations at the claim construction hearing held on May 24, 2018. For the sake of clarity, unless otherwise noted, the docket entries cited herein refer to the docket sheet in *Grecia v. MasterCard International, Inc.*, Case No. 15-cv-9059 (RJS) (S.D.N.Y.).



## II. LEGAL STANDARD

### A. Principles of Claim Construction

“[T]he claims of a patent define the invention to which the patentee is entitled the right to exclude.” *Aventis Pharm. Inc. v. Amino Chems. Ltd.*, 715 F.3d 1363, 1373 (Fed. Cir. 2013) (quoting *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (*en banc*)). Patent claim construction is a matter of law “exclusively within the province of the court.” *Markman*, 517 U.S. at 372. The terms of a claim generally take “their ordinary and customary meaning” – the meaning, that is, that the terms would have “to a person of ordinary skill in the art in question at the time of the invention.” *Phillips*, 415 F.3d at 1312–13. “A patentee, however, can act as his own lexicographer to specifically define terms of a claim contrary to their ordinary meaning.” *Abraxis Bioscience, Inc. v. Mayne Pharma (USA) Inc.*, 467 F.3d 1370, 1376 (Fed. Cir. 2006) (citation and internal quotation marks omitted). To act as his own lexicographer and use a term in a manner other than its ordinary meaning, the patentee must expressly define the term in the patent specification or the “prosecution history,” which consists of the record of the proceedings before the PTO. *Teleflex, Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002).

Courts consider the terms of a claim “in the context of the entire patent,” *Phillips*, 415 F.3d at 1313, and not “in a vacuum,” *Medrad, Inc. v. MRI Devices Corp.*, 401 F.3d 1313, 1319 (Fed. Cir. 2005) (citation and internal quotation marks omitted). Accordingly, courts construing patent claims examine the full “intrinsic evidence of record”: “the claims, the specification and, if in evidence, the prosecution history.” *PC Connector Solutions LLC v. SmartDisk Corp.*, 406 F.3d 1359, 1362 (Fed. Cir. 2005) (citation and internal quotation marks omitted). The specification in particular is “always highly relevant to the claim construction analysis” and is, in fact, “the single best guide to the meaning of a disputed term.” *Vitrionics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996). When “using the specification to interpret the meaning of a claim,”

however, courts should avoid “importing limitations from the specification into the claim.” *Phillips*, 415 F.3d at 1323; *see also Comark Commc'ns, Inc. v. Harris Corp.*, 156 F.3d 1182, 1186 (Fed. Cir. 1998) (“[T]here is sometimes a fine line between reading a claim in light of the specification, and reading a limitation into the claim from the specification.”). Likewise, while courts may consult the prosecution history to “determine whether . . . there were any express representations made in obtaining the patent regarding the scope and meaning of the claims,” *DeMarini Sports, Inc. v. Worth, Inc.*, 239 F.3d 1314, 1323 (Fed. Cir. 2001), the Federal Circuit has cautioned against relying too heavily on the prosecution history “because [it] represents an ongoing negotiation between the PTO and the applicant . . . [and] often lacks the clarity of the specification and thus is less useful for claim construction purposes,” *Phillips*, 415 F.3d at 1317.

In addition to considering such intrinsic evidence, courts construing patent claims may also consult extrinsic evidence, which includes “all evidence external to the patent and prosecution history, including expert and inventor testimony, dictionaries, and learned treatises.” *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 980 (Fed. Cir. 1995) (*en banc*), *aff'd*, 517 U.S. 370 (1996). Extrinsic evidence, however, is “less significant than the intrinsic record in determining the ‘legally operative meaning of claim language.’” *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 862 (Fed. Cir. 2004) (quoting *Vanderlande Indus. Nederland BV v. Int'l Trade Comm'n*, 366 F.3d 1311, 1318 (Fed. Cir. 2004)); *see also Phillips*, 415 F.3d at 1322–23 (“[J]udges are free to consult dictionaries and technical treatises . . . in order to better understand the underlying technology and may also rely on dictionary definitions when construing claim terms, so long as the dictionary definition does not contradict any definition found in or ascertained by a reading of the patent documents.”) (citation and internal quotation marks omitted); *Phillips*, 415 F.3d at 1318 (“[E]xpert testimony can be useful to a court . . . to provide background on the technology at issue



... or to establish that a particular term in the patent or the prior art has a particular meaning in the pertinent field.”). In general, if the meaning of the claim is clear from the intrinsic evidence alone, resort to extrinsic evidence is improper. *Vitrionics*, 90 F.3d at 1583 (“In most situations, an analysis of the intrinsic evidence alone will resolve any ambiguity in a disputed claim term. In such circumstances, it is improper to rely on extrinsic evidence.”)

B. Definiteness Under 35 U.S.C. § 112, ¶ 2<sup>3</sup>

Claims and claim terms must meet the patent law’s definiteness requirement. *See* 35 U.S.C. § 112, ¶ 2 (requiring that claims “particularly point[] out and distinctly claim[] the subject matter” of the invention). “Because claims delineate the patentee’s right to exclude,” the definiteness requirement ensures that the claims are “sufficiently definite to inform the public of the bounds of the protected invention.” *Halliburton Energy Servs., Inc. v. M-I LLC*, 514 F.3d 1244, 1249 (Fed. Cir. 2008). “[A] patent is invalid for indefiniteness if its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2124 (2014). In evaluating a claim for indefiniteness, courts must be mindful of the “inherent limitations of language,” and understand that “[s]ome modicum of uncertainty ... is the price of ensuring the appropriate incentives for innovation.” *Id.* at 2128 (internal quotation marks omitted). Furthermore, since issued patents are presumptively valid, *see*

---

<sup>3</sup> The parties do not appear to dispute that, because the first patent in the series was filed February 15, 2012 – prior to the enactment of the America Invents Act – the pre-AIA version of Section 112 is the one that applies here. (*See* Def. Br. at 4 & n.3.) When “the applications resulting in the patents at issue ... were filed before September 16, 2012, courts refer to the pre-AIA version of [Section] 112.” *AbbVie Deutschland GmbH & Co., KG v. Janssen Biotech, Inc.*, 759 F.3d 1285, 1290 (Fed. Cir. 2014). Here, although the applications for the ‘555 and ‘860 patents were filed *after* September 16, 2012, they are continuations of the ‘555 patent, which was filed several months *before* that date. In any event, the AIA does not appear to have materially modified the relevant language in the second paragraph of Section 112.

35 U.S.C. § 282, a party seeking to invalidate a claim as indefinite must do so by clear and convincing evidence. *See Microsoft Corp. v. i4i Ltd. P'ship*, 564 U.S. 91, 102 (2011).

Because “[a] determination of claim indefiniteness is a legal conclusion that is drawn from the court’s performance of its duty as the construer of patent claims,” *Personalized Medic. Commc’ns, LLC v. Int’l Trade Comm’n*, 161 F.3d 696, 705 (Fed. Cir. 1998), district courts may address indefiniteness at the claim construction stage rather than the summary judgment stage, *ePlus, Inc. v. Lawson Software, Inc.*, 700 F.3d 509, 517 (Fed. Cir. 2012) (“[I]ndefiniteness is a question of law and in effect part of claim construction.”); *see also Indus. Tech. Research Inst. v. LG Electronics Inc.*, No. 13-cv-02016 (GPC), 2014 WL 6907449, at \*3 (S.D. Cal. Dec. 8, 2014) (reviewing Federal Circuit case law on the timing of indefiniteness claims and concluding that district courts have “discretion as to when to determine indefiniteness during patent case proceedings”).

#### C. Means-Plus-Function Limitations Under 35 U.S.C. § 112, ¶ 6

Under paragraph six of Section 112, “[a]n element in a claim . . . may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.” When a claim term is drafted in a manner that falls within that provision, courts refer to the claim term a “means-plus-function limitation.” *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1347 (Fed. Cir. 2015). In enacting this provision, “Congress struck a balance in allowing patentees to express a claim limitation by reciting a function to be performed rather than by reciting structure for performing that function, while placing specific constraints on how such a limitation is to be construed, namely, by restricting the scope of coverage to only the structure, materials, or acts described in the specification as corresponding to the claimed function and equivalents thereof.” *Id.* at 1348.



The threshold question is whether a particular claim term is drafted in means-plus-function format. The presence of the word “means” creates a presumption in favor of construing a term as a means-plus-function limitation, and the absence of the word “means” raises the opposite presumption. *Id.* However, the Federal Circuit has emphasized that “the essential inquiry is not merely the presence or absence of the word ‘means’ but whether the words of the claim are understood by persons of ordinary skill in the art to have a sufficiently definite meaning as the name for structure.” *Id.* For example, “[g]eneric terms such as ‘mechanism,’ ‘element,’ ‘device,’” and other placeholder words that “reflect nothing more than verbal constructs” may be used in a claim “in a manner that is tantamount to using the word ‘means’ because ‘they typically do not connote sufficiently definite structure’ and therefore may invoke § 112, ¶ 6.” *Id.* (quoting *Mass. Inst. of Tech. & Elecs. for Imaging, Inc. v. Abacus Software*, 462 F.3d 1344, 1354 (Fed. Cir. 2006)).

Once a claim term has been identified as a means-plus-function limitation, construing it is a two-step process. First, the court identifies the claimed function. *Noah Sys., Inc. v. Intuit Inc.*, 675 F.3d 1302, 1311 (Fed. Cir. 2012). Then, the court determines what structure disclosed in the specification, if any, corresponds to the claimed function. *Id.* “Under this second step, structure disclosed in the specification is ‘corresponding’ structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim.” *Med. Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, 1210 (Fed. Cir. 2003) (citation and internal quotation marks omitted). If a patent fails to disclose adequate structure for performing the claimed function, the claim is invalid for indefiniteness under 35 U.S.C. § 112, ¶ 2. *Noah Sys.*, 675 F.3d at 1311–12; *Cardiac Pacemakers, Inc. v. St. Jude Med., Inc.*, 296 F.3d 1106, 1114 (Fed. Cir. 2002). Under paragraphs two and six of Section 112, therefore, “a means-plus-function clause is indefinite if a person of ordinary skill in the art would be unable to recognize

the structure in the specification and associate it with the corresponding function in the claim.” *Noah Sys.*, 675 F.3d at 1311 (citation and internal quotation marks omitted).

### III. DISCUSSION

The patents-in-suit relate to the field of digital rights management (“DRM”). DRM is a generic term for technologies used to prevent unauthorized copying or distribution of digital media, such as movies, music, or games, by restricting access to the media across devices, such as computers, phones, or tablets. (*See* ‘860 Patent at 1:19-22.) Grecia’s claimed invention “teaches a more personal system of digital rights management” that employs “electronic ID . . . to manage access rights across a plurality of devices.” (*Id.* at 1:23-26.)

According to the patents-in-suit, traditional DRM systems suffer from two notable shortcomings. First, traditional systems rely on digital content providers to maintain computer servers to receive and send authorization keys. (*Id.* at 2:54-62.) Sometimes content providers discontinue servers or go out of business after the DRM-encrypted content has been sold to consumers, who then lose the ability to access content they had purchased. (*Id.*) Second, traditional systems often function by linking rights to access content to the machine on which the content was acquired, which means that users can lose access to the content if the machine breaks or is stolen. (*Id.* at 2:1-8, 63-67). Grecia’s DRM system allegedly solves these problems by branding the information necessary to associate a user with the digital content into the metadata of the digital content, thereby enabling “unlimited interoperability between devices.” (*Id.* at 3:1-7, 4:2-9.)

The parties dispute the construction of 14 terms or sets of terms in the patents. The Court will address each term or set of terms in turn. Because the parties’ briefing postdates their joint claim construction chart and generally discusses the disputed terms in the sequence and grouping



adopted by Defendants in their claim construction brief, the Court will address the terms in that order.

A. “metadata of the digital content”<sup>4</sup>

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
Metadata of the digital content (‘860 Patent, Claims 9 and 21) (‘555 Patent, Claims 1, 12, and 15)	data that describes the digital content and is stored with the digital content	a data store

As Defendants point out, the term “metadata” is expressly defined in the specification. (‘860 Patent at 13:21-23 (“[M]etadata is defined simply as to ‘describe other data.’ It provides information about [a] certain item’s content.”).) For example, the metadata of a text document might include information such as the length of the document, when it was written, and who the author was. (*Id.* at 13:26-32.) In prior litigation involving the ‘860 Patent, Grecia himself took the position that “metadata of the digital content” means “data about the digital content.” (Doc. No. 68-5 at P006711.)

Nevertheless, Grecia now argues that “metadata of the digital content” should be construed as “a data store” because Claim 21 of the ‘860 Patent refers to “metadata of the digital content” as “being one or more of a database or storage in connection to the computer product.” (‘860 Patent at 17:64-67.) At the claim construction hearing, Grecia conceded that the term “metadata” does not ordinarily mean “a data store,” and suggested that he departed from that ordinary meaning by acting as his own lexicographer in the patents-in-suit. (Doc. No. 87 at 22.) However, there is a “heavy presumption that claim terms are to be given their ordinary and customary meaning.”

---

<sup>4</sup> As discussed below, *see infra* III.B, the parties appear to agree that the terms “digital content,” “cloud digital content,” and “encrypted digital media” are synonymous for relevant purposes.

*Aventis Pharm.*, 715 F.3d at 1373. Although patentees can define terms in unconventional ways, “[t]he standards for finding lexicography . . . are ‘exacting.’” *Rovi Guides, Inc. v. Comcast Corp.*, No. 16-cv-9278 (JPO), 2017 WL 3447989, at \*2 (S.D.N.Y. Aug. 10, 2017) (quoting *GE Lighting Solutions LLC v. AgiLight, Inc.*, 750 F.3d 1304, 1309 (Fed. Cir. 2014)). To act as his own lexicographer, the patentee must “clearly set forth a definition of the disputed claim term,” and “clearly express an intent to define the term” in the specification or the prosecution history. *Id.* at \*3 (citations and internal quotation marks omitted). Here, Grecia expressly defined the term “metadata” as data that describes other data, not as a storage device or repository for data. That distinction is reinforced by multiple diagrams in the specification that depict “database” and “metadata” as distinct components. (See ‘860 Patent, Figs. 3, 4.) Thus, “metadata of the digital content” is, at minimum, data that describes the digital content.

Defendants argue that the data describing the digital content must also be “stored with the digital content” because the specification describes the invention as involving a “branding action of at least one writable metadata as part of at least one digital media” (*id.* at 3:36-38). But logically, parts of a whole can be stored separately from other parts that comprise the whole. And Defendants do not appear to dispute that, as understood by a person of ordinary skill in the art, information such as the length of a document would still be metadata regardless of whether it is stored in the same file as the document. Nor does the patents’ definition of the term “metadata” reveal any locational restriction. (See ‘860 Patent at 13:21-23.) Moreover, to the extent that the intrinsic record leaves any ambiguity as to whether metadata is necessarily stored with the data it describes, that interpretation finds no support in dictionary definitions of the term. See, e.g., “Metadata,” *American Heritage Dictionary* (5th ed. 2011) (“Data that describes other data[.]”) And in proceedings before the PTO, MasterCard’s own expert opined that “it is well known in the art that



metadata need not be in the [same] media file as the digital content.” (Doc. No. 68-6 at 22.) Accordingly, the Court construes “metadata of the digital content” as simply “data that describes the digital content.”

B. “encrypted digital media”; “digital content”; “cloud digital content”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
(1) encrypted digital media; (2) digital content; (3) cloud digital content (‘860 Patent, Claims 9, 21, 22, 25) (‘555 Patent, Claims 1–3, 7, 8, 10–13, 15, 16, 17, 23–25) (‘308 Patent, Claim 1)	one or more of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated	data capable of being processed by a computer

The parties appear to agree that these three terms are synonymous for relevant purposes. Defendants nevertheless argue that because a key purpose of Grecia’s invention is to protect copyrights, these terms should be limited to types of media susceptible to copyright violations. But the specification expressly states that “[e]xamples of the encrypted digital files *include, and are not limited to*, a video file, an audio file, container formats, documents, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.” (‘860 Patent at 7:20-24 (emphasis added).) The examples are clearly illustrative, not exhaustive. Thus, the Court rejects Defendants’ artificially narrow construction and adopts Grecia’s proposed construction.

C. “verified web service”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
verified web service (‘860 Patent, Claims 9, 21) (‘555 Patent, Claims 1, 12, 15)	a web service that is used to authenticate the identity of a user or device	a machine-to-machine connected web service with an Applications Programmable Interface (API) that requires an

('308 Patent, Claim 1)		apparatus authentic identification
------------------------	--	------------------------------------

The term “verified web service” is mentioned only once in the specification:

The web service equipped with the API is usually a well-known membership-themed application in which the users must use an authentic identification. Some example[s] include[] Facebook in which as a rule, members are required to use their legal name identities. A reference number or name with the Facebook Platform API represents this information. Other *verified web services* in which real member names are required such as the LinkedIn API and the PayPal API and even others could be used, but for this discussion, Facebook will be used only as an example of how the authentication element of the invention is utilized.

(‘860 Patent at 10:41-51) (emphasis added). This description supports Defendants’ characterization of the verified web service as a “web service that is used to authenticate the identity of a user or device.” Although Grecia argues that a “machine-to-machine connect[ion]” requirement can be implied from the claim language, the Court disagrees; the claims refer to the role of the verified web service in facilitating the authentication process at a high level of generality, and shed little additional light on the structure of the web service. (*See, e.g.*, ‘860 Patent at 18:18-26.) Moreover, the Court sees no reason to embed the phrase “with an Application Programmable Interface (API)” into the definition of the term, given that this would duplicate the claim language. (*See id.* at 18:18-19) (“wherein the API is obtained from a verified web service”). Accordingly, the Court adopts Defendants’ proposed construction.

D. (1) “corresponding to the digital content”; (2) “corresponding to the digital media”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
(1) corresponding to the digital content; (2) corresponding to the encrypted digital media	modifies the claim term “[membership] verification token.”	no construction necessary



(‘860 Patent, Claims 9, 21)		
(‘555 Patent, Claims 1, 12, 15)		

The parties appear to agree that these terms are synonymous for relevant purposes. And Grecia acknowledges that these terms modify the term “verification token” in the relevant claims. (See Grecia Br. at 7.) Thus, the Court deems the parties to have stipulated to Defendants’ proposed construction. See *VDP Patent, LLC v. Welch Allyn Holdings, Inc.*, 623 F. Supp. 2d 414, 426 (S.D.N.Y. 2008) (“These elements need not be construed by this Court, as the parties have stipulated to their meaning.”)

E. (i) “authorizing access to . . .” through (v) “user access rights associated to . . .”

<b>Term</b>	<b>Defendants’ Proposed Construction(s)</b>	<b>Grecia’s Proposed Construction</b>
(i) authorizing access to digital content (‘860 Patent, Claims 9, 21)	(i) and (ii):	no construction necessary
(ii) monitoring access to an encrypted digital media (‘555 Patent, Claims 1, 12, 15)	providing infinite access rights to digital content to a first user or content acquirer and recognized friends and family of the content acquirer or first user	
(iii) facilitating interoperability between a plurality of data processing devices (‘555 Patent, Claims 1, 12, 15)	(iii), (iv), and (v):	
(iv) facilitating access rights between a plurality of data processing devices (‘860 Patent, Claim 9)	providing infinite access rights to a first user’s or content acquirer’s multiple compatible devices	
(v) user access rights associated to the cloud digital content (‘308 Patent, Claim 1)		

The parties appear to agree that term (i) is synonymous with term (ii) and that terms (iii) through (v) are synonymous with each other. Defendants argue that these terms necessarily contemplate “infinite access rights” because the specification describes the invention as “directed

at providing infinite access rights . . . to the content acquirer . . . and optionally to their recognized friends and family.” (‘860 Patent at 5:7-12.) Just because the invention is *generally aimed at* providing access rights of infinite duration, however, does not mean that the duration is necessarily infinite in every conceivable embodiment of the invention. The same goes for Defendants’ proposed limitation regarding “friends and family” of the content acquirer. Although friends and family may be among the most likely recipients of the access rights, nothing in the patents limits the potential recipients to people who share intimate social relations or bloodlines with the primary user. To adopt Defendants’ constructions would be to commit the error of “importing limitations from the specification into the claim.” *Phillips*, 415 F.3d at 1323. Accordingly, the Court agrees with Grecia that no construction is necessary.

F. “verification token”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
verification token (‘860 Patent, Claims 9, 10, 21, 26) (‘555 Patent, Claims 1, 2, 10–12, 15–17, 24, 25) (‘308 Patent, Claim 1)	MasterCard: no construction necessary  Samsung: data that represents a permission to access [digital media / digital content / cloud digital content] and that is different from the identification reference	data that represents permission to access [digital media / digital content / cloud digital content]

Although Grecia’s and Samsung’s proposed constructions of this term are largely identical, MasterCard disputes their characterization of the “verification token” as “permission” or “a permission” rather than “a request for permission,” and takes the position that no construction is necessary. (Def. Br. at 14.) However, according to the specification, the verification token “represents permission from the content provider to grant access rights to the [user.]” (‘860 Patent at 9:21-23.) That description supports the constructions favored by Samsung and Grecia. Grecia’s



analogy to a “hall pass” is instructive (*see* Grecia Reply Br. at 7); although it is true that the verification token must eventually be authenticated (*see* ‘860 Patent at 3:61-64), the same is often true of passes and permits – which are typically thought of as representing permission, not merely a request for permission. Furthermore, as to the latter half of Samsung’s proposed construction, Grecia agrees that the “verification token” is different from the “identification reference,” but argues that this distinction is clear enough and requires no construction. The Court agrees; indeed, Samsung offers no explanation as to why anyone would think the two terms might refer to the same thing. Accordingly, the Court adopts Grecia’s proposed construction.

G. “identification reference”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
identification reference (‘860 Patent, Claims 9 and 21) (‘555 Patent, Claims 1, 12 and 15)	an identifier of a web service, that can be used to identify a user of the web service or a device that accesses the web service, and is different from the verification token	no construction necessary

Defendants assert that the term “identification reference,” as used in the patents-in-suit, means “an identifier of a web service, that can be used to identify a user of the web service or a device that accesses the web service, and is different from the verification token.” (Def. Br. at 15.) But the specification and the claim language indicate a broader meaning. To be sure, the specification illustrates the term “identification reference” with the example of a user or device identifier associated with a web service membership, such as a serial number corresponding to a Facebook user’s name. (*See* ‘860 Patent at 11:18-24.) In at least one specified embodiment of the invention, however, the “identification reference” is requested from a “communications console” that is only “*usually*” connected to the Internet and only “*usually . . .* part of a web service.” (*Id.*

at 3:41-45 (emphasis added).) Moreover, in Claim 21 of the ‘860 Patent, the “identification reference” is described broadly as comprising “one or more of a verified web account identifier, letter, number, rights token, e-mail, password, access time, serial number, manufacturer identification, checksum, operating system version, browser version, credential cookie or key, or ID.” (*Id.* at 18:24-30.) Given these indications that the “identification reference” may not always be associated with a web service, the Court construes the term to mean simply “an identifier that can be used to identify a user or device.”

H. “communications console”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
communications console (‘860 Patent, Claims 9 and 21) (‘555 Patent, Claims 1, 12 and 15)	a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API)  ‘860 Patent, Claim 21 construction only: a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) separate from the memory, CPU, and non-transitory computer usable medium of the computer product	no construction necessary

In every claim in which the term “communications console” appears, it is described as “a combination of a graphic user interface (GUI) and an Applications Programmable Interface.” (*See* ‘555 Patent at 14:49-52, 16:17-19, 16:62-65; ‘860 Patent at 16:8-10, 18:15-17.) Indeed, Grecia does not dispute Defendants’ interpretation of the term. Rather, Grecia argues that construction is unnecessary because the term is clearly and consistently defined in the claims. Because the Court agrees with Grecia that construction is unnecessary – including with respect to whether the



communications console is “separate from the memory, CPU, and non-transitory computer usable medium” described in Claim 21 of the ‘860 Patent, which is readily apparent from the claim itself – the Court declines to construe this term.

I. “credential assigned to the apparatus of (a)”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
credential assigned to the apparatus of (a) (‘308 Patent, Claim 1)	any data that represents a permission to conduct a data exchange session	no construction necessary

The term “credential assigned to the apparatus of (a)” appears only in Claim 1 of the ‘308 Patent, which purports to describe “a process for transforming a user access request for cloud digital content into a computer readable authorization object[.]” (‘360 Patent at 14:31-33.) Step (c) in the process involves “establishing an API communication between the apparatus of (a) and a database apparatus . . . wherein establishing the API communication requires *a credential assigned to the apparatus of (a)*, wherein the apparatus assigned credential is recognized as a permission to conduct a data exchange session between the apparatus of (a) and the database apparatus to complete the verification process[.]” (*Id.* at 14:48-58 (emphasis added).) In light of this description, Defendants’ proposed construction – “any data that represents permission to conduct a data exchange session” – would create redundancies by duplicating the claim language. Nor do Defendants explain why their construction incorporates the first part of the clause following the term, but not the second part (“between the apparatus of (a) . . .”). Accordingly, the Court agrees with Grecia that construction is unnecessary: “credential assigned to the apparatus of (a)” means “credential assigned to the apparatus of (a).”

## J. “apparatus of (a)”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
apparatus of (a) (‘308 Patent, Claim 1)	a user device and any hardware, software, or web application running on the user device that sends the access request for the cloud digital content	no construction necessary

Similarly, the term “apparatus of (a)” appears only in Claim 1 of the ‘308 Patent. Defendants contend that “it is clear from the context of the claim language” that the “apparatus of (a)” is “located on the user device side, rather than the server side.” (Def. Br. at 16.) The Court disagrees. Step (a) of the process described in Claim 1 involves, in relevant part, “receiving an access request for cloud digital content through *an apparatus* in process with at least one CPU . . . wherein the access request further comprises verification data provided by at least one user, wherein the verification data is recognized by *the apparatus* as a verification token.” (‘308 Patent at 14:34-44 (emphasis added).) Subsequent sections of Claim 1 refer back to that apparatus as the “apparatus of (a).” (*See, e.g., id.* at 14:48-49.) Thus, all that is clear from the claim language is that the “apparatus of (a)” is “in process with at least one CPU” and that it is capable of recognizing certain data as a verification token. (*Id.* at 14:34-44.) Moreover, Defendants’ proposed construction implies that the apparatus is a tangible device, whereas in the specification, the word “apparatus” is also used to refer to software or web applications. (*See, e.g.,* ‘308 Patent at 6:59-61, 10:51-56, 11:8-11.) In any event, the term “apparatus of (a)” clearly refers to the apparatus mentioned in step (a) of the process described in Claim 1. Accordingly, the Court agrees with Grecia that no construction is necessary.



## K. “computer product configured to perform. . .”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
computer product configured to perform the steps of . . . (i) . . . receiving a digital content access request from the communications console . . . (ii) . . . authenticating the verification token; (iii) . . . establishing a connection with the communications console; (iv) . . . requesting the at least one identification reference from the at least one communications console; (v) . . . receiving the at least one identification reference from the communications console (vi) . . . writing at least one of the verification token or the identification reference into the said metadata (‘860 Patent, Claim 21)	Means-plus-function term subject to the requirements of 35 U.S.C. § 112, ¶ 6, that is indefinite for failure to disclose corresponding structure in the specification	no construction necessary

Defendants argue that Claim 21 of the ‘860 Patent must be construed pursuant to 35 U.S.C. § 112, ¶ 6 because, they contend, even though it lacks the word “means,” it is drafted in means-plus-function format. Where, as here, a claim term lacks the word “means,” there is a rebuttable presumption that it is not a means-plus-function limitation. *Williamson*, 792 F.3d at 1348. Nevertheless, the presumption can be overcome if the challenger demonstrates that the claim term “fails to recite sufficiently definite structure or else recites function without reciting sufficient structure for performing that function.” *Id.* In *Williamson*, the Federal Circuit held that a claim that attributed several functions to a “distributed learning control module” was drafted in means-plus-function format. *Id.* at 1331.

This claim presents a closer question for two reasons. First, it uses the term “computer product,” instead of a “well-known nonce word” such as “module,” “mechanism,” “element,” or “device.” *Id.* at 1350. But the term “computer product,” especially in our digital age, is hardly much more specific than “module” or “device”; it is essentially a “black box.” *See Verint Sys. Inc. v. Red Box Recorders Ltd.*, 166 F. Supp. 3d 364, 372 (S.D.N.Y. 2016) (applying § 112, ¶ 6 to the phrase “a first computer application operative to access the voice information”). Second, whereas the claim in *Williamson* did not identify specific hardware components, *see* 792 F.3d at 1350, the claim here specifies that the computer product comprises “a memory, a CPU, a communications console and a non-transitory computer usable medium, the computer usable medium having an operating system stored therein.” (‘860 Patent at 17:52-55.) But these generic components clearly do not provide “sufficient structure for performing” the claimed functions. *Williamson*, 792 F.3d at 1348; *see also id.* at 1351 (“[T]he fact that one of skill in the art could program a computer to perform the recited functions cannot create structure where none is otherwise disclosed.”).

Grecia insists that Claim 21 “does not invoke the rules applied to means-plus-function claiming because, as in [*Apple Inc. v. Motorola, Inc.*, 757 F.3d 1286 (Fed. Cir. 2014)], [C]laim 21 ‘describes its operation, including its input, output, and how its output may be achieved.’” (Grecia Reply Br. at 10.) The claim at issue in *Apple*, however, provided a much more detailed explanation of how its functions could be achieved, including “the initial angle of a finger contact, the number of fingers making contact, a specific swiping gesture, [tapping] a certain location on the screen or the angle of movement of a finger on the screen.” 757 F.3d at 1303; *see also Williamson*, 792 F.3d at 1351) (“While portions of the claim do describe certain inputs and outputs at a very high level (*e.g.*, communications between the presenter and audience member computer systems), the claim does not . . . otherwise impart structure to the “distributed learning control module” as recited in



the claim.”) *Apple* is also distinguishable because it was decided when the Federal Circuit was still applying a “strong” presumption against construing a term as a means-plus-function limitation in the absence of the word “means.” The Federal Circuit has since abandoned that approach. *See Williamson*, 792 F.3d at 1349 (“Our consideration of this case has led us to conclude that such a heightened burden is unjustified and that we should abandon characterizing as ‘strong’ the presumption that a limitation lacking the word ‘means’ is not subject to § 112, para. 6.”). Here, the Court finds that Defendants have rebutted the presumption by showing that Claim 21 does not recite sufficient structure for performing the claimed functions. Accordingly, Claim 21 is subject to the rigors of Section 112.

If a claim is subject to paragraph six of Section 112, it must provide sufficient “corresponding structure” in the specification. *Med. Instrumentation*, 344 F.3d at 1210 (quoting 35 U.S.C. § ¶ 112, ¶ 6). “[S]tructure disclosed in the specification is ‘corresponding’ structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim.” *Id.* (citation and internal quotation marks omitted). If a patent fails to disclose adequate structure for performing the claimed function, the claim is invalid for indefiniteness under Section 112. *Noah Sys.*, 675 F.3d at 1311.

Here, the specification fails to clarify how Claim 21’s generic components are “configured” to achieve the identified functions. In his claim construction briefs, Grecia cites various broad swathes of the specification and states in conclusory fashion that the “computer science is . . . thoroughly detailed” therein. (*See Grecia Reply Br.* at 11.) In fact, however, the computer science in the specification is anything but thorough or detailed. Although the specification contains a vague description of the components of a generic computer (*see* ‘860 Patent at 5:25-32), in cases such as this, where the claim obviously “must be implemented in a . . .

computer programmed to perform particular functions pursuant to instructions from program software,” the Federal Circuit has consistently “require[d] that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor.” *Williamson*, 792 F.3d at 1353 (collecting cases). Rather, the specification must “disclose an algorithm for performing the claimed function.” *Id.* Here, the specification fails to identify any specialized hardware, software, or algorithms that are “clearly link[ed]” to the functions recited in Claim 21. *Med. Instrumentation*, 344 F.3d at 1210. Thus, as Defendants argue, Claim 21 – and its dependent claims, Claims 22–30 (*see* Doc. No. 65-2) – are each invalid as indefinite under Section 112. *See Williamson*, 792 F.3d at 1354; *see also Interval Licensing LLC v. AOL, Inc.*, 766 F.3d 1364, 1374 (Fed. Cir. 2014) (finding claim indefinite for including an indefinite specification).

L. (1) “two-way data exchange session”; (2) “to complete a verification process”

Defendants state that they “are no longer seeking construction of these two terms” (Def. Br. 21), and Grecia agrees that no construction is necessary. Accordingly, the Court declines to construe them.

M. (1) “first receipt module” . . . (7) “customization module”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
(1) “first receipt module”; (2) “authentication module”; (3) “connection module”; (4) “request module”; (5) “secondary receipt module”; (6) “branding module”; (‘860 Patent, Claim 9) (‘555 Patent, Claims 12–14) (7) “customization module”	Means-plus-function term subject to the requirements of 35 U.S.C. § 112, ¶ 6, that is indefinite for failure to disclose corresponding structure in the specification	(1) receives a computer data access write or read request from at least one communications console of the plurality of data processing devices (2) authenticates the data that represents a permission to access digital content (3) establishes a connection with the at least one communications console (4) requests at least one electronic identification reference



('860 Patent, Claims 21, 24)		from the at least one communications console (5) receives the at least one electronic identification reference from at least one communications console (6) writes to the data store associated with the data capable of being processed by a computer (7) no construction necessary
------------------------------	--	---

As Defendants argue, these terms fall squarely within the Federal Circuit's mandate that "module" terms are generally "means-plus-function" limitations subject to Section 112. *See Williamson*, 792 F.3d at 1350–51. "'Module' is a well-known nonce word that can operate as a substitute for 'means.'" *Id.* at 1350. "[T]he word 'module' does not provide any indication of structure because it sets forth the the same black box recitation of structure . . . as if the term 'means' had been used." *Id.* Here, as in *Williamson*, the claims of the patents-in-suit do not specify any structure for the "module" terms, but define them solely by their purported functions. Indeed, the "constructions" Grecia proposed in the Joint Claim Construction Chart merely paraphrase those claimed functions, which are substantially identical to the functions of the "computer product" terms. (*See* Doc. No. 62-1 at 8, 23–28.) Thus, there can be little doubt that the "module" terms are means-plus-functions limitations subject to Section 112.

Moreover, as set forth above with respect to the similar "computer product" terms, the specification fails to "clearly link or associate" sufficient corresponding structure for performing the "module" functions. As with the "computer product" terms, Grecia merely cites broad swathes of the specification and contends that structure is somehow disclosed therein for each module. (Grecia Reply Br. at 12–13.) For example, Grecia asserts in conclusory fashion that structures for the "first receipt module" and the "authentication module" are taught at Figures 3, 4, and 5 of the

specification and columns 8 through 10 of the specification, without specifying what those structures are. (*See id.* (citing ‘860 Patent at 8:52-10:2, Figs. 3–5).) As Defendants point out, the portions of the specification cited by Grecia merely describe various aspects of creating digital tokens, storing the tokens, and using the tokens to set up royalty payments; they do not explain how a “first receipt module” supposedly “receiv[es] a digital content access request” from a “communications console” or how the “authentication module” purportedly “authentica[tes] the verification token.” (‘860 Patent at 15:56-58, 16:4-5.) Grecia’s arguments with respect to the other modules are similarly cursory, imprecise, and unilluminating. (*See* Grecia Reply Br. at 12–13.) Thus, the Court concludes that the specification fails to disclose sufficient structure and “clearly link” it to the functions recited in Claims, 9, 21, and 24 of the ‘860 Patent and Claims 12, 13, and 14 of the ‘555 Patent. *Williamson*, 792 F.3d at 1354. Accordingly, those claims and their dependent claims (e.g., Claims 24–26 of the ‘555 Patent, which are dependent on Claim 12, and Claims 22–30 of the ‘860 Patent, which are dependent on Claim 21) are each invalid as indefinite under Section 112. *See id.* at 1354; *see also Interval Licensing LLC*, 766 F.3d at 1374.

N. “the metadata of the digital content being one or more of a database or storage”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
the metadata of the digital content being one or more of a database or storage (‘860 Patent, Claim 21)	indefinite under 35 U.S.C. § 112, ¶ 2	no construction necessary

Defendants argue that this term is indefinite under paragraph two of Section 112. “[A] patent must be precise enough to afford clear notice of what is claimed, thereby apprising the public of what is still open to them.” *Nautilus*, 134 S. Ct. at 2129 (citation, alterations, and internal quotation marks omitted). “[A] patent is invalid for indefiniteness if its claims, read in light of the



specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Id.* at 2124. Here, because the specification expressly defines “metadata” as data that describes other data, *see supra* Section III.C, the Court agrees with Defendants that this term, which conflates metadata with a database or repository, makes no sense and is therefore indefinite – akin to the phrase, “the car being one or more of a parking lot or garage.” Indeed, Grecia does not appear to dispute that if, as the Court has now found, “metadata of the digital content” means “data that describes the digital content,” this term is indefinite; rather, Grecia reiterates his argument, which the Court has now rejected, that “metadata of the digital content” means “a data store.” Thus, the Court concludes this term is invalid as indefinite under § 112, ¶ 2.

O. (1) “the verification token is handled . . . ” (2) “the access request is a request . . . ”

Term	Defendants’ Proposed Construction(s)	Grecia’s Proposed Construction
(1) the verification token is handled by the user as a redeemable instrument (‘860 Patent, Claim 21)	indefinite under 35 U.S.C. § 112, ¶ 2	no construction necessary
(2) the access request is a request from a first user, the first user is a human user in operation of the computer product and establishes first access to the digital content; or wherein the access request is a request from a secondary user, the secondary user is a human user in operation of the computer product and establishes secondary access to the same digital content as first established for access by the first user (‘860 Patent, Claim 22)		

As a rule, a single claim that covers both an apparatus and a method of using that apparatus is indefinite under Section 112. *IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377, 1384 (Fed. Cir. 2005). Such mixed claims fail to clarify “whether infringement . . . occurs when one creates a system that allows the user to [perform the step] . . . or . . . when the user actually [performs the step].” *Id.* Defendants argue that Claims 21 and 22 of the ‘860 Patent are mixed claims because, Defendants contend, the claims cover both an apparatus (a “computer product”) and methods of using that apparatus.

Claim 21 describes a “computer product” that is “configured to perform” six steps. The phrase identified by Defendants appears in the following context:

the computer product configured to perform the steps of . . . receiving the digital content access request . . . the request comprising a verification token corresponding to the digital content, *the verification token is handled by the user as a redeemable instrument*, wherein the verification token comprises at least one of a purchase permission, a rental permission, or a membership permission[.]”

(‘860 Patent at 17:60–18:6 (emphasis added)). It is clear from this context that the phrase “handled by the user as a redeemable instrument” describes the type of verification token at issue, not an independent step performed by the user. “Unlike *IPXL* and similar cases, the claim[] at issue here makes clear that the . . . limitation reflects the capability of that structure rather than the activities of the user.” *UltimatePointer, L.L.C. v. Nintendo Co.*, 816 F.3d 816, 827 (Fed. Cir. 2016). Thus, the Court rejects Defendants’ argument that Claim 21 is a mixed method-and-apparatus claim.

Claim 22 covers “[t]he computer product according to [C]laim 21, wherein the access request is a request from a first user, the first user is a human user in operation of the computer product and establishes first access to the digital content; or wherein the access request is a request



from a secondary user, the secondary user is a human user in operation of the computer product and establishes secondary access to the same digital contest as first established for access by the first user.” (‘860 Patent at 18:34-43.) Defendants argue that this is a mixed claim because it “purports to claim an apparatus, *i.e.*, a computer product” but “require[s] action performed by users of the computer product.” (Def. Br. 25.) But not every claim that contemplates interaction between a user and an apparatus is a mixed claim. *See MasterMine Software, Inc. v. Microsoft Corp.*, 874 F.3d 1307, 1316 (Fed. Cir. 2017). As with Claim 21, the Court finds that Claim 22 clearly covers the computer product, not a particular method of using that product. Accordingly, the Court rejects Defendants’ argument that Claim 22 is indefinite as a mixed claim.<sup>5</sup>

#### IV. CONCLUSION

For the reasons stated above, the Court adopts the constructions set forth in this opinion and finds that the following claims are invalid for indefiniteness:

- Patent No. 8,402,555: Claims 12–14 and 24–26
- Patent No. 8,533,860: Claims 9, 10, and 21–30

With respect to the remaining claims, per the Court’s case management plan and scheduling order (Doc. No. 48), all fact discovery must be completed no later than 60 days after the entry of this claim construction order. Accordingly, IT IS HEREBY ORDERED THAT (1) the parties shall file any pre-motion letters by November 5, 2018; and (2) the parties shall appear for a post-discovery conference on November 20, 2018 at 11:00 a.m. If either party submits a

---

<sup>5</sup> Nonetheless, for reasons already discussed above, the Court has determined that Claim 21 – and its dependent claims, including Claim 22 – are indefinite on other grounds. *See infra* Parts III.K, III.M, III.N.

timely pre-motion letter, the post-discovery conference shall also serve as a pre-motion conference.

SO ORDERED.

Dated: September 7, 2018  
New York, New York

  
\_\_\_\_\_  
RICHARD J. SULLIVAN  
United States District Judge

# EXHIBIT 13

MODIFIED PTO/SB/01 (06-07)

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

# DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION (37 CFR 1.63)

Declaration  
Submitted  
With Initial  
Filing

OR

Declaration  
Submitted after Initial  
FilingAttorney Docket  
Number

B7-1

First Named Inventor

William Grecia

COMPLETE IF KNOWN

Application Number

Not Yet Assigned

Filing Date

Herewith

Art Unit

Not Yet Assigned

Examiner Name

Not Yet Assigned

I hereby declare that:

Each inventor's residence, mailing address, and citizenship are as stated below next to their name.

I believe the inventor(s) named below to be the original and first inventor(s) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)

(Title of the invention)

the specification of which



is attached hereto

OR



was filed on (MM/DD/YYYY)

as United States Application Number or PCT International

Application Number

and was amended on (MM/DD/YYYY)

(if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56, including for continuation-in-part applications, material information which became available between the filing date of the prior application and the national or PCT international filing date of the continuation-in-part application.

I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or (f), or 365(b) of any foreign application(s) for patent, inventor's or plant breeder's rights certificate(s), or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent, inventor's or plant breeder's rights certificate(s), or any PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached?	
				YES	NO
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

☐ Additional foreign application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.

[Page 1 of 2]

This collection of information is required by 35 U.S.C. 115 and 37 CFR 1.63. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 21 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance completing the form, call 1-800-PTO-9199 and select option 2.



Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

**DECLARATION —[Utility or Design Patent Application]**Direct all  
correspondence to:The address  
associated with  
Customer Number:

000070984

OR

Correspondence  
address below

Name

Address

City

State

ZIP

Country

Telephone

Email

**WARNING:**

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**NAME OF SOLE OR FIRST INVENTOR:**

A petition has been filed for this unsigned inventor

Given Name (first and middle (if any))

Family Name or Surname

William

Grecia

Inventor's Signature

Date

/william gracia/

02/15/2012

Residence: City

State

Country

Citizenship

Grandville

Michigan

USA

USA

Mailing Address

2885 Sanford Avenue, Southwest, #13208

City

State

Zip

Country

Grandville

Michigan

49418

USA



Additional inventors or a legal representative are being named on the supplemental sheet(s) PTO/SB/02A or 02LR attached hereto.

# EXHIBIT 14

## UNITED STATES PATENT AND TRADEMARK OFFICE CERTIFICATE OF CORRECTION

Page 1 of 1

PATENT NO. : 8,402,555  
 APPLICATION NO.: 13/397,517  
 ISSUE DATE : 03-19-2013  
 INVENTOR(S) : William Grecia

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In claim 1, column 14, line 49; claim 12, column 16, line 17; and claim 15, column 16, line 63:

the word "connection", in each occurrence, should be changed to --a connection--

In claim 1, column 14, line 52; claim 12, column 16, line 20 and 21; and claim 15, column 16, line 66:

the word "Applications", in each occurrence, should be changed to --Application--

In claim 1, column 14, line 53; claim 12, column 16, line 21; and claim 15, column 16, line 67:

the phrase "obtained from", in each occurrence, should be changed to --related to--

In claim 14, column 16, line 43:

the word "on" should be changed to --one--

In claim 19, column 17, line 47:

the word "-comprising" should read --comprising--

### MAILING ADDRESS OF SENDER (Please do not use customer number below):

William Grecia  
 2885 Sanford Ave SW #13208  
 Grandville, MI 49418

This collection of information is required by 37 CFR 1.322, 1.323, and 1.324. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1.0 hour to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Attention Certificate of Corrections Branch, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

*If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.*

## Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

William Grecia

Patent No.: 8,402,555

Issued: March 19, 2013

Examiner: Tran, Tri

Art Unit: 2494

CNF# 6106

**COCR**

Assistant Commissioner for Patents

P. O. Box 1450

Alexandria VA, 22313-1450

**Request for Certificate of Correction under 37 C.F.R. § 1.323**

This paper requests a Certificate of Correction under 37 C.F.R. § 1.323 for the United States patent identified above. Accompanying this request is a Certificate of Correction form PTO/SB/44 containing the text of this correction. Applicant submits the fee due to corrections being clerical, typographical, and of minor character.

Respectfully,

/william grecia/

William Grecia

Applicant Pro Se

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	13397517			
<b>Filing Date:</b>	15-Feb-2012			
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)			
<b>First Named Inventor/Applicant Name:</b>	William Grecia			
<b>Filer:</b>	William Grecia			
<b>Attorney Docket Number:</b>	B7-1			
Filed as Small Entity				
<b>Utility under 35 USC 111(a) Filing Fees</b>				
Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
Certificate of Correction	1811	1	100	100
<b>Extension-of-Time:</b>				



Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Miscellaneous:				
Total in USD (\$)				100

**Electronic Acknowledgement Receipt**

<b>EFS ID:</b>	16612009
<b>Application Number:</b>	13397517
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	6106
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)
<b>First Named Inventor/Applicant Name:</b>	William Grecia
<b>Customer Number:</b>	70984
<b>Filer:</b>	William Grecia
<b>Filer Authorized By:</b>	
<b>Attorney Docket Number:</b>	B7-1
<b>Receipt Date:</b>	16-AUG-2013
<b>Filing Date:</b>	15-FEB-2012
<b>Time Stamp:</b>	14:41:37
<b>Application Type:</b>	Utility under 35 USC 111(a)

**Payment information:**

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	\$ 100
RAM confirmation Number	975
Deposit Account	
Authorized User	

**File Listing:**

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
-----------------	----------------------	-----------	-------------------------------------	------------------	------------------

1	Request for Certificate of Correction	sb0044-555patent-c.pdf	164456 d571a534514f06890069928d9ed342afa77 efeb6	no	2
<b>Warnings:</b>					
<b>Information:</b>					
2	Request for Certificate of Correction	COC.pdf	205476 61b4953668ff268aa354885d6dce109918d 71862	no	1
<b>Warnings:</b>					
<b>Information:</b>					
3	Fee Worksheet (SB06)	fee-info.pdf	29939 8bed7211c3b82aeb3ed2965e6e36990ff87 4567d	no	2
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			399871		
<p><b>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</b></p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b>  <b>If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</b></p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b>  <b>If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</b></p> <p><b><u>New International Application Filed with the USPTO as a Receiving Office</u></b>  <b>If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</b></p>					

# EXHIBIT 15

**UNITED STATES PATENT AND TRADEMARK OFFICE  
CERTIFICATE OF CORRECTION**Page 1 of 1

PATENT NO. : 8,402,555  
APPLICATION NO.: 13/397,517  
ISSUE DATE : 03-19-2013  
INVENTOR(S) : William Grecia

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In claim 11, column 15, line 57; claim 16, column 17, line 14; and claim 25, column 18, line 26:

"selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by",

in each occurrence, should be changed to and read as

--selected from the group consisting of a purchase permission, a rental permission, and a membership permission, coupled to a royalty scheme; wherein the purchase permission, rental permission, and membership permission is represented by--

In claim 4, column 15, line 16; and claim 20, column 17, line 52:

"content provider" in each occurrence, should be changed to and read as --a content provider--

**MAILING ADDRESS OF SENDER (Please do not use customer number below):**

William Grecia  
2885 Sanford Ave SW #13208  
Grandville, MI 49418

This collection of information is required by 37 CFR 1.322, 1.323, and 1.324. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1.0 hour to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Attention Certificate of Corrections Branch, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

*If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.*

## Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

William Grecia  
Patent No.: 8,402,555  
Issued: 03-19-2013

Examiner: Tran, Tri  
Art Unit: 2494

**COCR**

Assistant Commissioner for Patents  
P. O. Box 1450  
Alexandria VA, 22313-1450

**Request for Certificate of Correction under 37 C.F.R. § 1.323**

This paper requests a second Certificate of Correction under 37 C.F.R. § 1.323 for the United States patent identified above. Accompanying this request is a Certificate of Correction form PTO/SB/44 containing the text of this correction. Applicant submits the fee due to corrections being:

- (1) of a clerical nature,
- (2) of a typographical nature, or
- (3) a mistake of minor character.

This request does not involve changes that would:

- (1) constitute new matter or
- (2) require reexamination

Respectfully,  
/william grecia/  
William Grecia  
Applicant Pro Se

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	13397517			
<b>Filing Date:</b>	15-Feb-2012			
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)			
<b>First Named Inventor/Applicant Name:</b>	William Grecia			
<b>Filer:</b>	William Grecia			
<b>Attorney Docket Number:</b>	B7-1			
Filed as Small Entity				
<b>Utility under 35 USC 111(a) Filing Fees</b>				
Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
Certificate of Correction	1811	1	100	100
<b>Extension-of-Time:</b>				

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
<b>Miscellaneous:</b>				
<b>Total in USD (\$)</b>				<b>100</b>

**Electronic Acknowledgement Receipt**

<b>EFS ID:</b>	16858005
<b>Application Number:</b>	13397517
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	6106
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)
<b>First Named Inventor/Applicant Name:</b>	William Grecia
<b>Customer Number:</b>	70984
<b>Filer:</b>	William Grecia
<b>Filer Authorized By:</b>	
<b>Attorney Docket Number:</b>	B7-1
<b>Receipt Date:</b>	16-SEP-2013
<b>Filing Date:</b>	15-FEB-2012
<b>Time Stamp:</b>	10:52:28
<b>Application Type:</b>	Utility under 35 USC 111(a)

**Payment information:**

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	\$ 100
RAM confirmation Number	9733
Deposit Account	
Authorized User	

**File Listing:**

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
-----------------	----------------------	-----------	-------------------------------------	------------------	------------------

1	Request for Certificate of Correction	sb0044-555patent-c2.pdf	164829 2d7f31ec38f18b78e40ef90ad86a8c5566c4a14e	no	2
<b>Warnings:</b>					
<b>Information:</b>					
2	Request for Certificate of Correction	COC-555-2.pdf	106546 869fbb1ead0ddc3f1496199654f221146a59e901	no	1
<b>Warnings:</b>					
<b>Information:</b>					
3	Fee Worksheet (SB06)	fee-info.pdf	29939 c5ed33a13315a3fd77b938b9bb5af0a7dd1fdc9	no	2
<b>Warnings:</b>					
<b>Information:</b>					
<b>Total Files Size (in bytes):</b>			301314		
<p><b>This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.</b></p> <p><b><u>New Applications Under 35 U.S.C. 111</u></b>  <b>If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.</b></p> <p><b><u>National Stage of an International Application under 35 U.S.C. 371</u></b>  <b>If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.</b></p> <p><b><u>New International Application Filed with the USPTO as a Receiving Office</u></b>  <b>If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.</b></p>					

# EXHIBIT 16



## UNITED STATES PATENT AND TRADEMARK OFFICE CERTIFICATE OF CORRECTION

Page \_\_\_\_\_ of \_\_\_\_\_

PATENT NO. : 8,402,555  
APPLICATION NO.: 13/397,517  
ISSUE DATE : 03-19-2013  
INVENTOR(S) : William Grecia

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In claim 23, column 18, line 11:

the word "the" should be changed to --a--

### MAILING ADDRESS OF SENDER (Please do not use Customer Number below):

William Grecia  
2885 Sanford Ave SW #13208 Grandville, MI 49418

This collection of information is required by 37 CFR 1.322, 1.323, and 1.324. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1.0 hour to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Attention Certificate of Corrections Branch, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

*If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.*

## Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

William Grecia  
Patent No.: 8,402,555  
Issued: 03-19-2013

Examiner: Tran, Tri  
Art Unit: 2494

**COCR**

Assistant Commissioner for Patents  
P. O. Box 1450  
Alexandria VA, 22313-1450

**Request for Certificate of Correction under 37 C.F.R. § 1.323**

This paper requests a second Certificate of Correction under 37 C.F.R. § 1.323 for the United States patent identified above. Accompanying this request is a Certificate of Correction form PTO/SB/44 containing the text of this correction. Applicant submits the fee due to corrections being:

- (1) of a clerical nature,
- (2) of a typographical nature, or
- (3) a mistake of minor character.

This request does not involve changes that would:

- (1) constitute new matter or
- (2) require reexamination

Respectfully,  
/william grecia/  
William Grecia  
Applicant Pro Se

## Electronic Patent Application Fee Transmittal

<b>Application Number:</b>	13397517			
<b>Filing Date:</b>	15-Feb-2012			
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)			
<b>First Named Inventor/Applicant Name:</b>	William Grecia			
<b>Filer:</b>	William Grecia			
<b>Attorney Docket Number:</b>	B7-1			
Filed as Large Entity				
<b>Filing Fees for Utility under 35 USC 111(a)</b>				
<b>Description</b>	<b>Fee Code</b>	<b>Quantity</b>	<b>Amount</b>	<b>Sub-Total in USD(\$)</b>
<b>Basic Filing:</b>				
<b>Pages:</b>				
<b>Claims:</b>				
<b>Miscellaneous-Filing:</b>				
<b>Petition:</b>				
<b>Patent-Appeals-and-Interference:</b>				
<b>Post-Allowance-and-Post-Issuance:</b>				
CERTIFICATE OF CORRECTION	1811	1	160	160

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Extension-of-Time:				
Miscellaneous:				
Total in USD (\$)				160

**Electronic Acknowledgement Receipt**

<b>EFS ID:</b>	41475127
<b>Application Number:</b>	13397517
<b>International Application Number:</b>	
<b>Confirmation Number:</b>	6106
<b>Title of Invention:</b>	PERSONALIZED DIGITAL MEDIA ACCESS SYSTEM (PDMAS)
<b>First Named Inventor/Applicant Name:</b>	William Grecia
<b>Customer Number:</b>	70984
<b>Filer:</b>	William Grecia
<b>Filer Authorized By:</b>	
<b>Attorney Docket Number:</b>	B7-1
<b>Receipt Date:</b>	23-DEC-2020
<b>Filing Date:</b>	15-FEB-2012
<b>Time Stamp:</b>	11:17:11
<b>Application Type:</b>	Utility under 35 USC 111(a)

**Payment information:**

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	\$ 160
RAM confirmation Number	E2020BMB18284395
Deposit Account	
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:



**File Listing:**

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Certificate of Correction	sb0044.pdf	163017	no	2
			e15995cf321c5df336523a9c027313814c49bbbe		

**Warnings:****Information:**

2	Request for Certificate of Correction	COC-2020.pdf	17626	no	1
			8a33f5b70f65038e915dacc73349df3a5d2c5d9c		

**Warnings:****Information:**

3	Fee Worksheet (SB06)	fee-info.pdf	30418	no	2
			740790906723ff67aedc4c57f24da868f63a8d9c		

**Warnings:****Information:**

<b>Total Files Size (in bytes):</b>			211061
-------------------------------------	--	--	--------

**This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.**

**New Applications Under 35 U.S.C. 111**

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

**National Stage of an International Application under 35 U.S.C. 371**

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

**New International Application Filed with the USPTO as a Receiving Office**

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

# EXHIBIT 17

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,402,555 B2  
APPLICATION NO. : 13/397517  
DATED : March 19, 2013  
INVENTOR(S) : William Grecia

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims

In claim 1, column 14, line 49; claim 12, column 16, line 17; and claim 15, column 16, line 63:

the word "connection", in each occurrence, should be changed to --a connection--

In claim 1, column 14, line 52; claim 12, column 16, line 20 and 21; and claim 15, column 16, line 66:

the word "Applications", in each occurrence, should be changed to --Application--

In claim 1, column 14, line 53; claim 12, column 16, line 21; and claim 15, column 16, line 67:

the phrase "obtained from", in each occurrence, should be changed to --related to--

In claim 14, column 16, line 43:

the word "on" should be changed to --one--

In claim 19, column 17, line 47:

the word "-comprising" should read --comprising--

Signed and Sealed this  
Twenty-fourth Day of September, 2013



Teresa Stanek Rea  
*Deputy Director of the United States Patent and Trademark Office*

# EXHIBIT 18

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,402,555 B2  
APPLICATION NO. : 13/397517  
DATED : March 19, 2013  
INVENTOR(S) : William Grecia

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims

In claim 11, column 15, line 57; claim 16, column 17, line 14; and claim 25, column 18, line 26:

“selected from a group consisting of a purchase permission, a rental permission, or membership permission coupled to a royalty scheme; wherein the permission is represented by”,

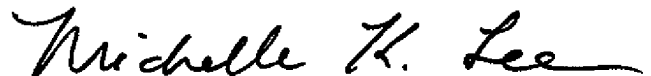
in each occurrence, should be changed to and read as

--selected from the group consisting of a purchase permission, a rental permission, and a membership permission, coupled to a royalty scheme; wherein the purchase permission, rental permission, and membership permission is represented by--

In claim 4, column 15, line 16; and claim 20, column 17, line 52:

“content provider” in each occurrence, should be changed to and read as --a content provider--

Signed and Sealed this  
Fourth Day of February, 2014



Michelle K. Lee  
*Deputy Director of the United States Patent and Trademark Office*

# EXHIBIT 19





US008887308B2

(12) **United States Patent**  
**Grecia**

(10) **Patent No.:** **US 8,887,308 B2**

(45) **Date of Patent:** **\*Nov. 11, 2014**

(54) **DIGITAL CLOUD ACCESS (PDMAS PART III)**

(71) Applicant: **William Grecia**, Downingtown, PA (US)

(72) Inventor: **William Grecia**, Downingtown, PA (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 75 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/888,051**

(22) Filed: **May 6, 2013**

(65) **Prior Publication Data**

US 2014/0304778 A1 Oct. 9, 2014

**Related U.S. Application Data**

(63) Continuation of application No. 13/740,086, filed on Jan. 11, 2013, now Pat. No. 8,533,860, which is a continuation of application No. 13/397,517, filed on Feb. 15, 2012, now Pat. No. 8,402,555, which is a continuation of application No. 12/985,351, filed on Jan. 6, 2011, now abandoned, which is a continuation of application No. 12/728,218, filed on Mar. 21, 2010, now abandoned.

(51) **Int. Cl.**  
**H04L 29/06** (2006.01)

(52) **U.S. Cl.**  
CPC ..... **H04L 63/10** (2013.01)  
USPC ..... **726/29; 726/28; 713/185**

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

7,254,235 B2	8/2007	Boudreault et al.	
7,343,014 B2	3/2008	Sovio et al.	
7,526,650 B1 *	4/2009	Wimmer	713/176
2005/0065891 A1	3/2005	Lee et al.	
2008/0010685 A1	1/2008	Holtzman et al.	
2009/0083541 A1	3/2009	Levine	
2010/0100899 A1	4/2010	Bradbury et al.	
2011/0288946 A1 *	11/2011	Baiya et al.	705/26.1

**OTHER PUBLICATIONS**

U.S. Appl. No. 13/397,517 Notice of Allowance.

U.S. Appl. No. 13/740,086 Notice of Allowance.

\* cited by examiner

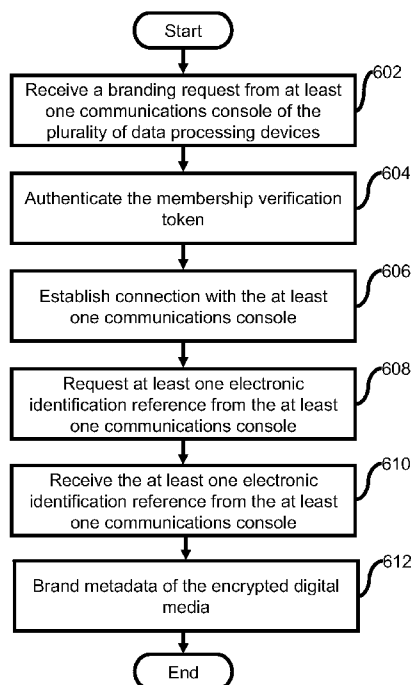
*Primary Examiner* — Jung Kim

*Assistant Examiner* — Tri Tran

(57) **ABSTRACT**

The invention is an apparatus that facilitates access to a data source to accept verification and authentication from an enabler using at least one token and at least one reference. The at least one reference could be a device serial number, a networking MAC address, or a membership ID reference from a web service. Access to the data source is also managed with a plurality of secondary enablers.

**1 Claim, 7 Drawing Sheets**



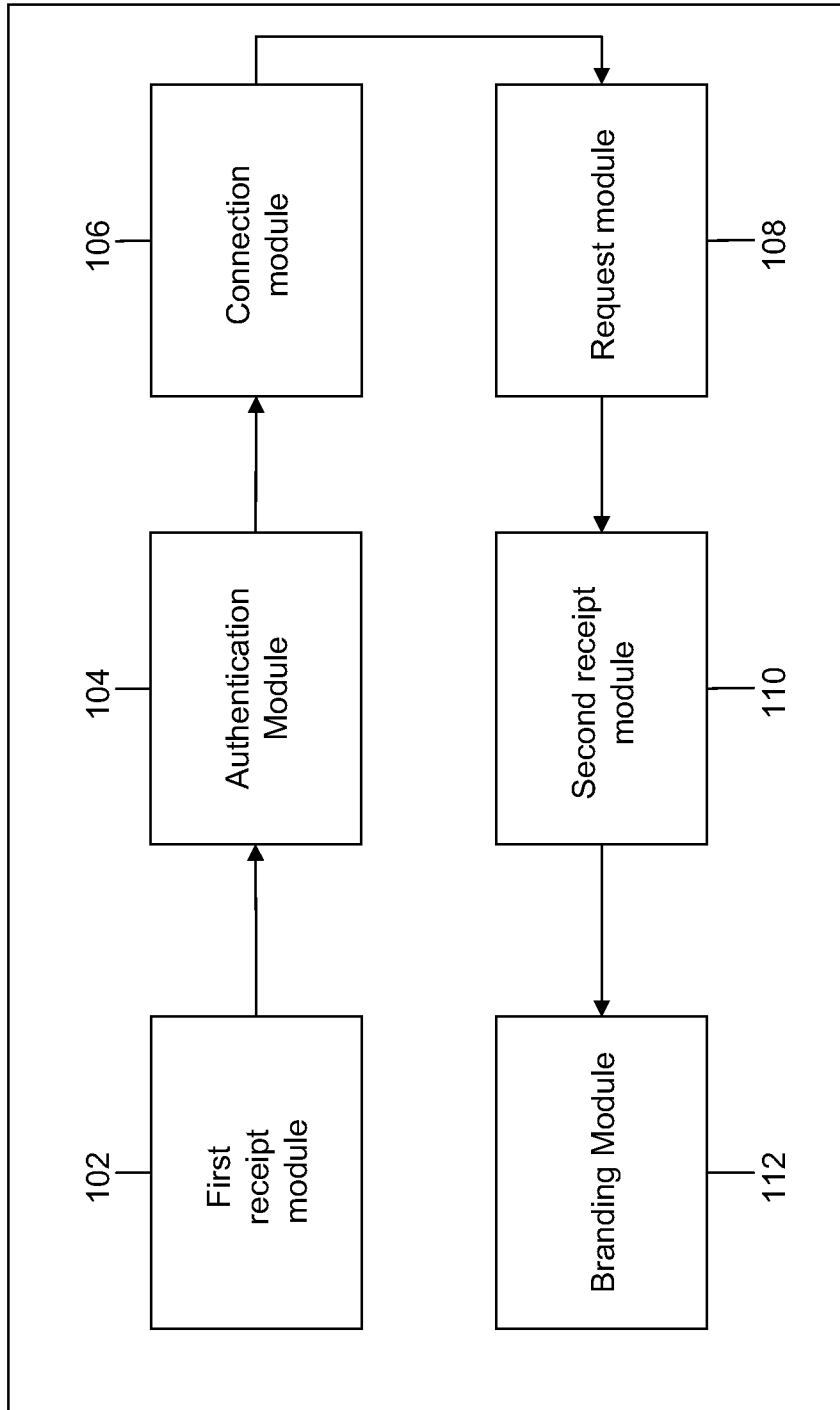


FIG.1

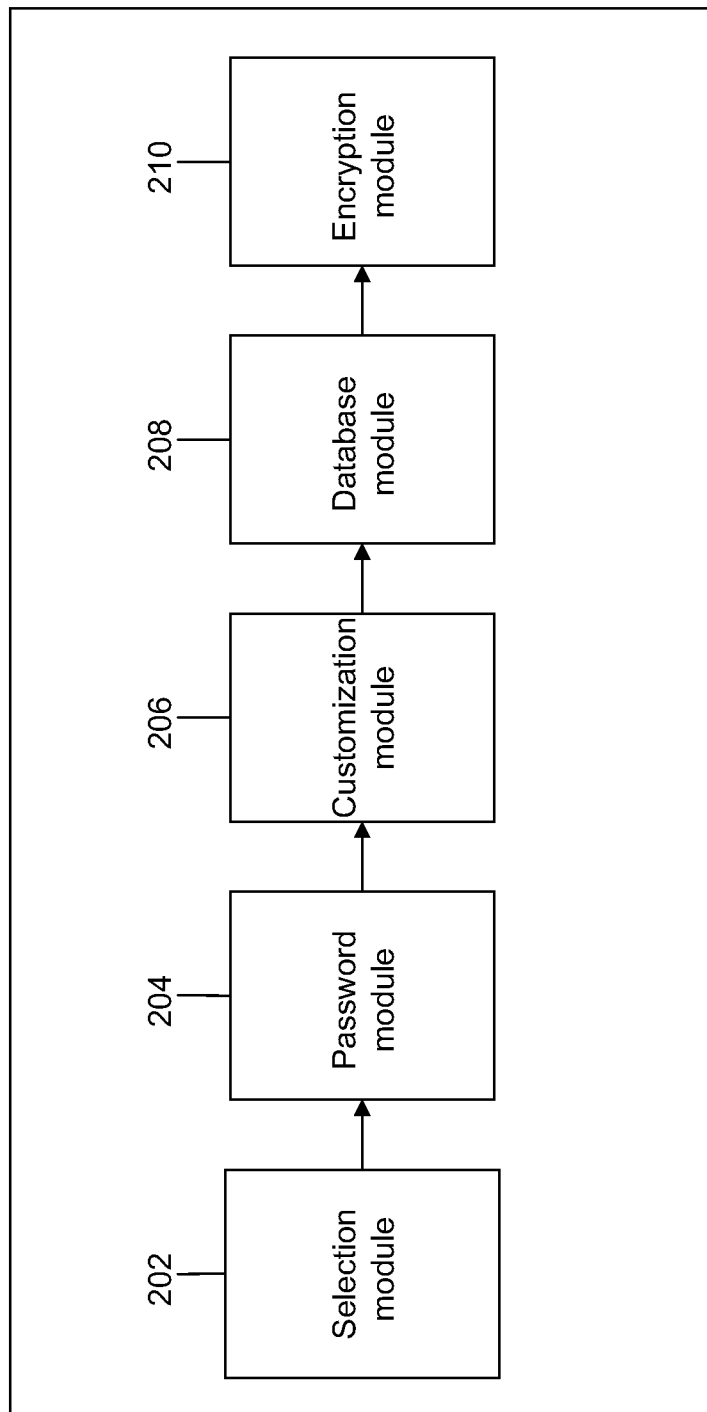


FIG.2

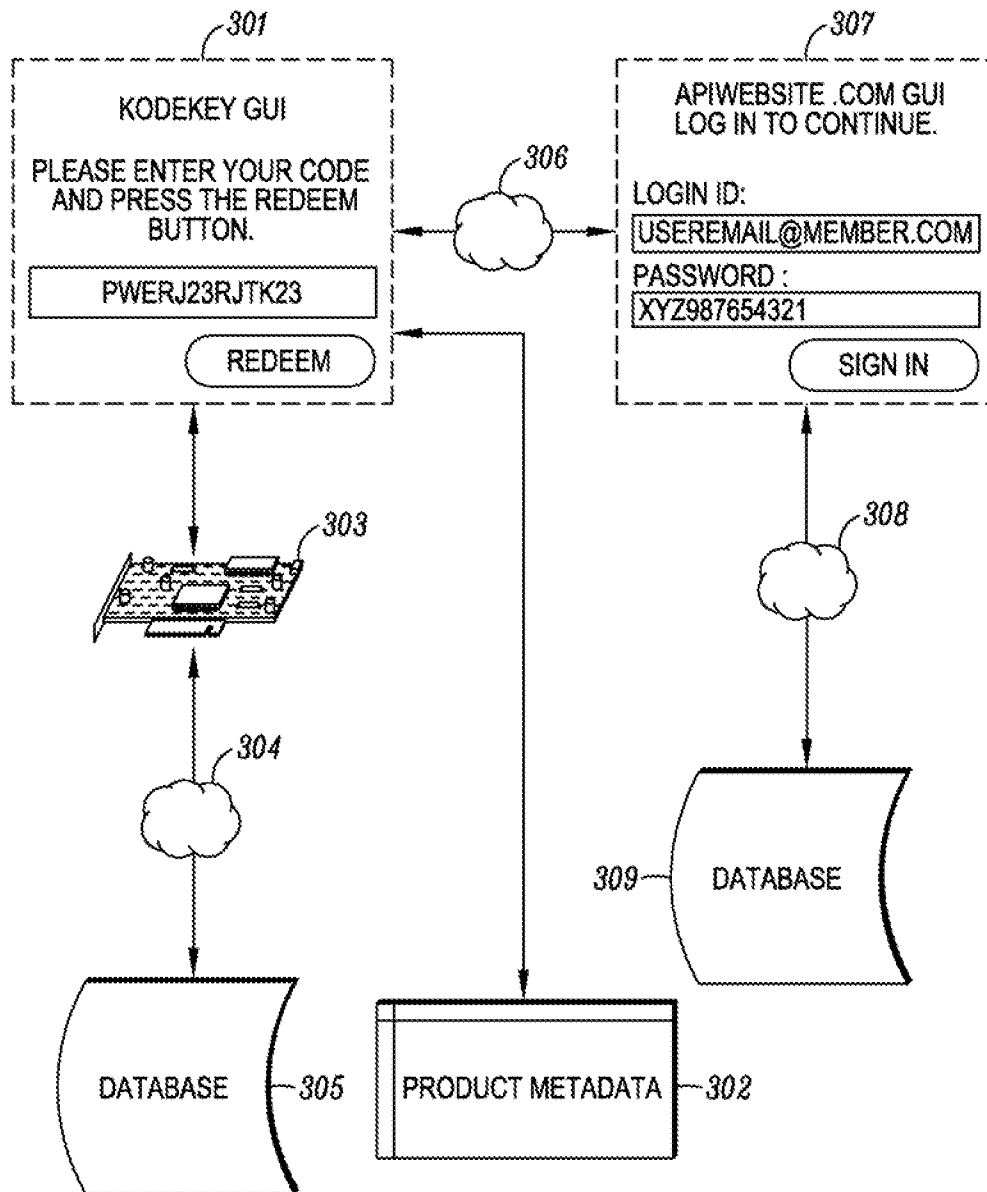


FIG. 3

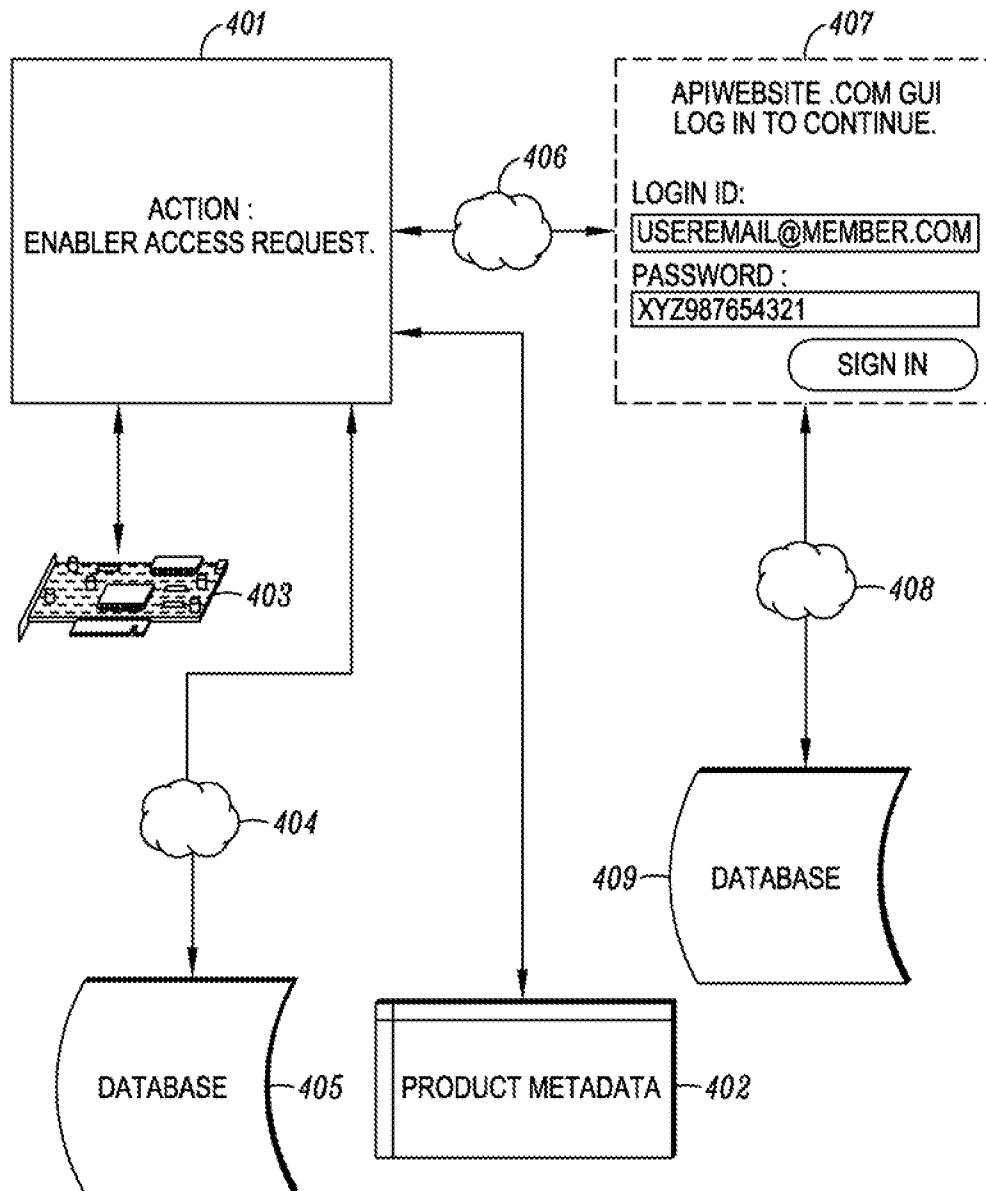


FIG. 4

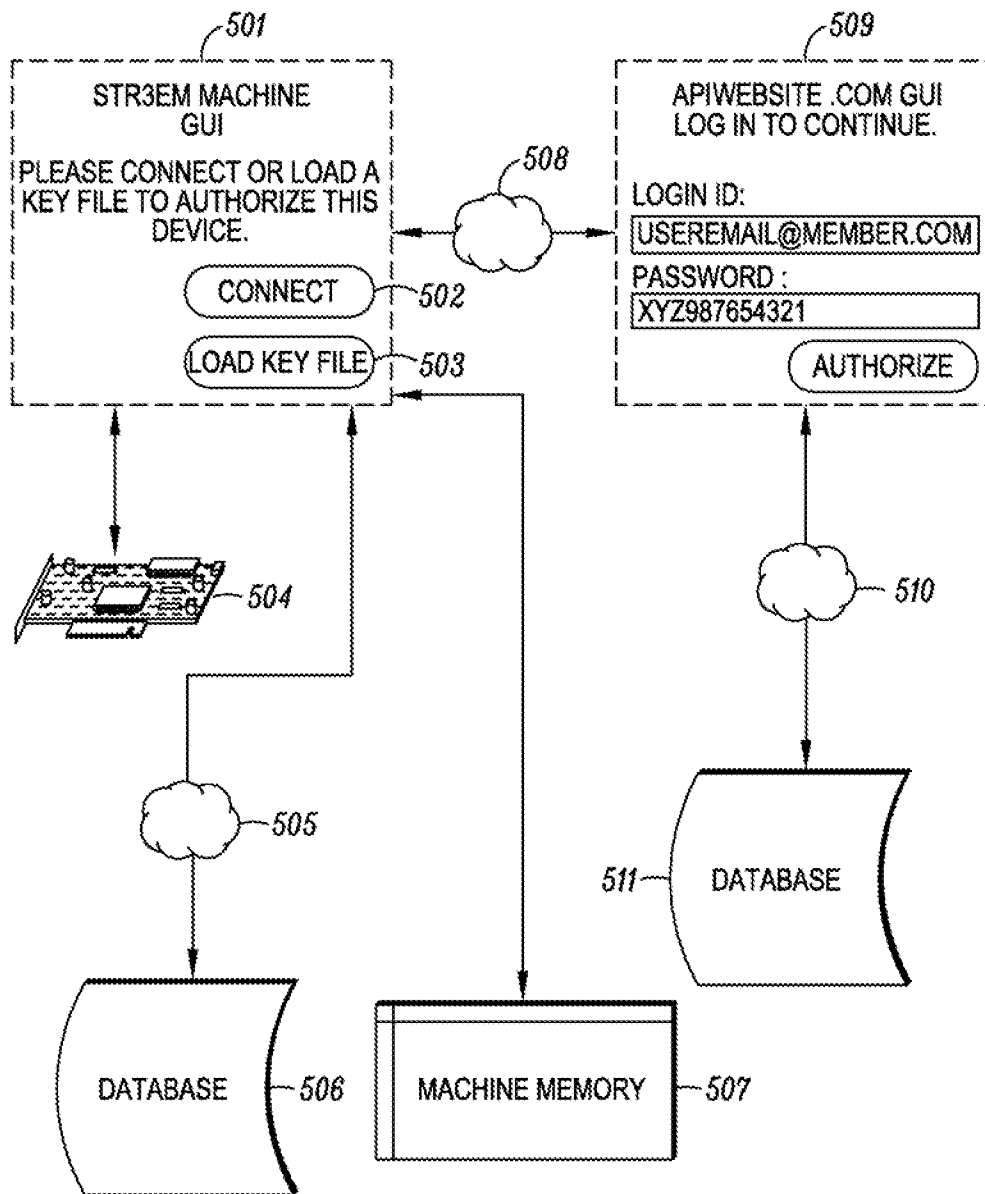


FIG. 5



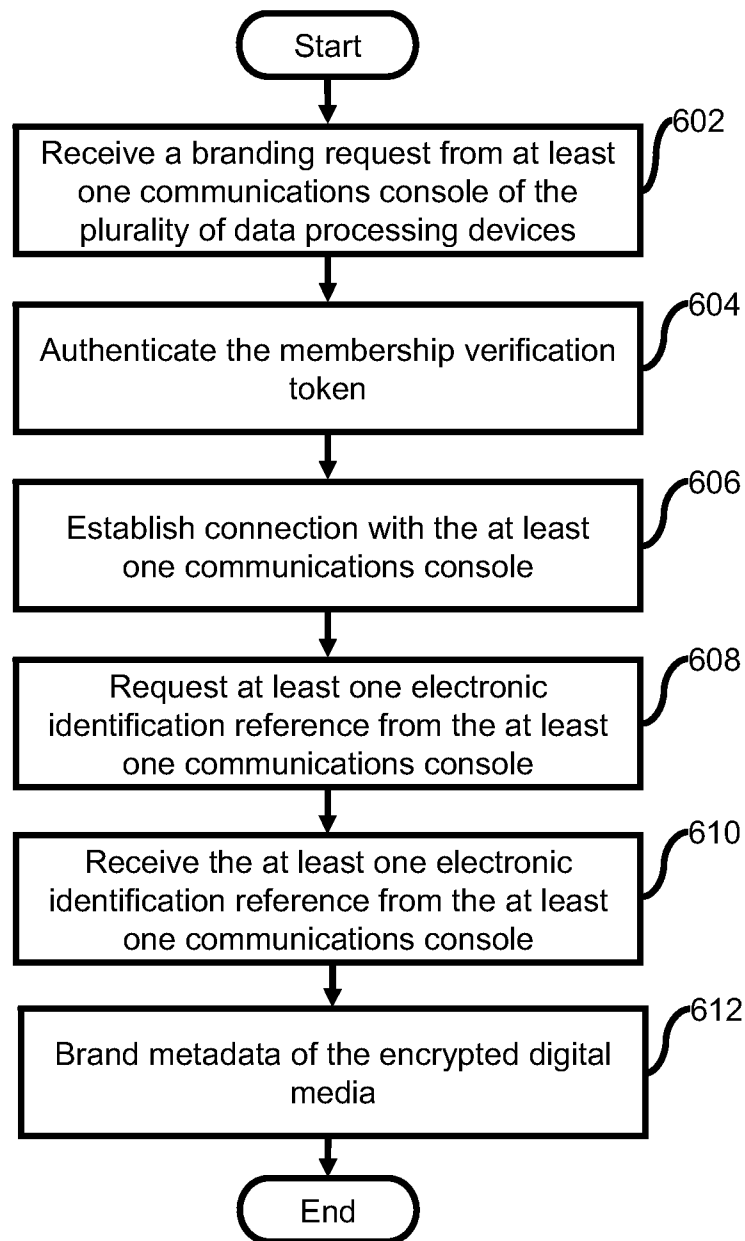


FIG.6

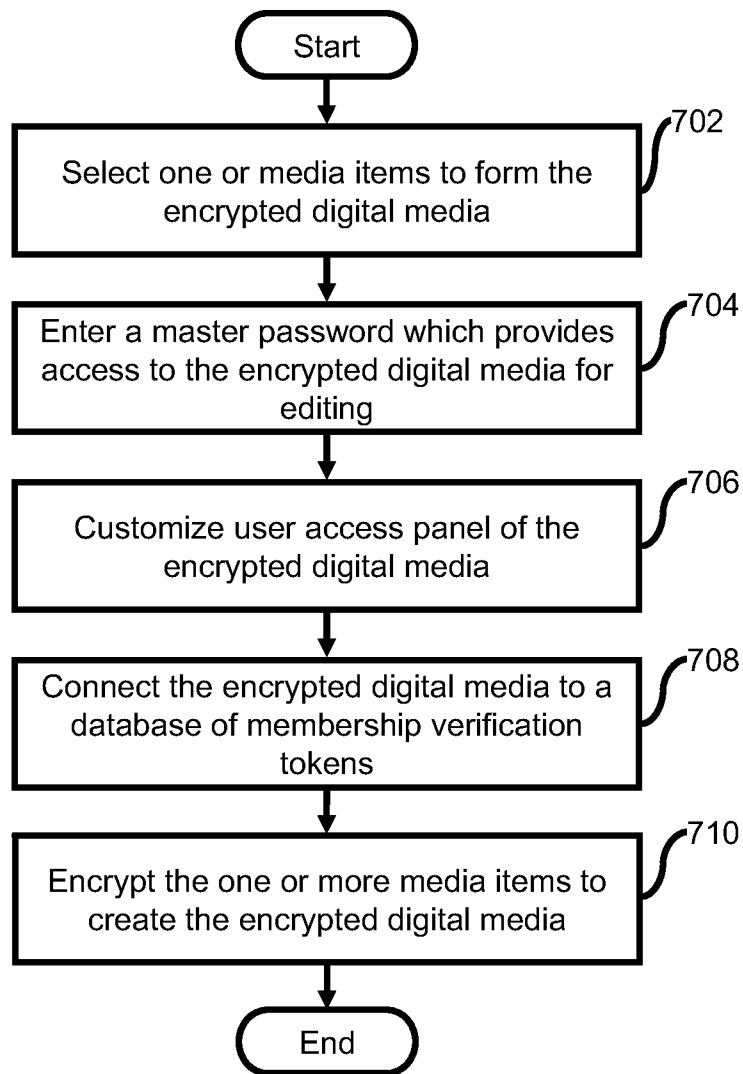


FIG.7

US 8,887,308 B2

1

**DIGITAL CLOUD ACCESS (PDMAS PART III)****CROSS-REFERENCE TO RELATED APPLICATIONS**

This application is a continuation of and claims the priority benefit of U.S. patent application Ser. No. 13/740,086 filed Jan. 11, 2013 which is a continuation of and claims the priority benefit of Ser. No. 13/397,517 filed Feb. 15, 2012 now issued as U.S. Pat. No. 8,402,555 on Mar. 19, 2013 which is a continuation of and claimed the priority benefit of Ser. No. 12/985,351 filed Jan. 6, 2011 which was a continuation of and claimed the priority benefit of U.S. patent application Ser. No. 12/728,218 filed Mar. 21, 2010, which are incorporated herein by reference in their entirety.

**BACKGROUND OF THE INVENTION****1. Field of the Invention**

The present invention relates to the field of digital rights management schemes used by creators of electronic products to protect commercial intellectual property copyrights privy to illegal copying using computerized devices. More specifically, the present invention teaches a more personal system of digital rights management which employs electronic ID, as part of a web service membership, to manage access rights across a plurality of devices.

**2. Description of the Prior Art**

Digital rights management (DRM) is a generic term for access control technologies used by hardware manufacturers, publishers, copyright holders and individuals to impose limitations on the usage of digital content across devices. DRM refers to any technology that inhibits undesirable or illegal uses of the digital content. The term generally doesn't refer to forms of copy protection that can be circumvented without modifying the file or device, such as serial numbers or key files. It can also refer to restrictions associated with specific instances of digital works or devices.

Traditional DRM schemes are defined as authentication components added to digital files that have been encrypted from public access. Encryption schemes are not DRM methods but DRM systems are implemented to use an additional layer of authentication in which permission is granted for access to the cipher key required to decrypt files for access. A computer server is established to host decryption keys and to accept authentication keys from Internet connected client computers running client software in which handles the encrypted files. The server can administer different authorization keys back to the client computer that can grant different sets of rules and a time frame granted before the client is required to connect with the server to reauthorize access permissions. In some cases content can terminate access after a set amount of time, or the process can break if the provider of the DRM server ever ceases to offer services.

In the present scenario, consumer entertainment industries are in the transition of delivering products on physical media such as CD and DVD to Internet delivered systems. The Compact Disc, introduced to the public in 1982, was initially designed as a proprietary system offering strict media to player compatibility. As the popularity of home computers and CD-ROM drives rose, so did the availability of CD ripping applications to make local copies of music to be enjoyed without the use of the disc. After a while, users found ways to share digital versions of music in the form of MP3 files that could be easily shared with family and friends over the Internet. The DVD format introduced in 1997 included a new apparatus for optical discs technology with embedded copy

2

protection schemes also recognized as an early form of DRM. With internet delivered music and video files, DRM schemes has been developed to lock acquired media to specific machines and most times limiting playback rights to a single machine or among a limited number of multiple machines regardless of the model number. This was achieved by writing the machine device ID to the metadata of the media file, then cross referencing with a trusted clearinghouse according to pre-set rules. DRM systems employed by DVD and CD technologies consisted of scrambling (also known as encryption) disc sectors in a pattern to which hardware developed to unscramble (also known as decryption) the disc sectors are required for playback. DRM systems built into operating systems such as Microsoft Windows Vista block viewing of media when an unsigned software application is running to prevent unauthorized copying of a media asset during playback. DRM used in computer games such as SecuROM and Steam are used to limit the amount of times a user can install a game on a machine. DRM schemes for e-books include embedding credit card information and other personal information inside the metadata area of a delivered file format and restricting the compatibility of the file with a limited number of reader devices and computer applications.

In a typical DRM system, a product is encrypted using Symmetric block ciphers such as DES and AES to provide high levels of security. Ciphers known as asymmetric or public key/private key systems are used to manage access to encrypted products. In asymmetric systems the key used to encrypt a product is not the same as that used to decrypt it. If a product has been encrypted using one key of a pair it cannot be decrypted even by someone else who has that key. Only the matching key of the pair can be used for decryption. After receiving an authorization token from a first-use action are usually triggers to decrypt block ciphers in most DRM systems. User rights and restrictions are established during this first-use action with the corresponding hosting device of a DRM protected product.

Examples of such prior DRM art include Hurtado (U.S. Pat. No. 6,611,812) who described a digital rights management system, where upon request to access digital content, encryption and decryption keys are exchanged and managed via an authenticity clearing house. Other examples include Alve (U.S. Pat. No. 7,568,111) who teaches a DRM and Tuoriniemi (U.S. Pat. No. 20090164776) who described a management scheme to control access to electronic content by recording use across a plurality of trustworthy devices that has been granted permission to work within the scheme.

Recently, DRM schemes have proven unpopular with consumers and rights organizations that oppose the complications with compatibility across machines manufactured by different companies. Reasons given to DRM opposition range from limited device playback restrictions to the loss of fair-use which defines the freedom to share media products will family members.

Prior art DRM methods rely on content providers to maintain computer servers to receive and send session authorization keys to client computers with an Internet connection. Usually rights are given from the server for an amount of time or amount of access actions before a requirement to reconnect with the server is required for reauthorization. At times, content providers will discontinue servers or even go out of business some years after DRM encrypted content was sold to consumers causing the ability to access files to terminate.

In the light of the foregoing discussion, the current states of DRM measures are not satisfactory because unavoidable issues can arise such as hardware failure or property theft that could lead to a paying customer loosing the right to recover

US 8,887,308 B2

3

purchased products. The current metadata writable DRM measures do not offer a way to provide unlimited interoperability between different machines. Therefore, a solution is needed to give consumers the unlimited interoperability between devices and “fair use” sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments.

#### SUMMARY OF THE INVENTION

An object of the present invention is to provide unlimited interoperability of digital media between unlimited machines with management of end-user access to the digital media.

In accordance with an embodiment of the present invention, the invention is a process of an apparatus which in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods (herein referred to as The App) is used to: handle at least one branding action which could include post read and write requests of at least one writable metadata as part of at least one digital media asset to identify and manage requests from at least one excelsior enabler, and can further identify and manage requests from a plurality of connected second enablers; with at least one token and at least one electronic identification reference received from the at least one excelsior enabler utilizing at least one membership. Here, controlled by the at least one excelsior enabler, The App will proceed to receive the at least one token to verify the authenticity of the branding action and further requests; then establish at least one connection with at least one programmable communications console of the at least one membership to request and receive the at least one electronic identification reference; and could request and receive other data information from the at least one membership. The method then involves sending and receiving variable data information from The App to the at least one membership to verify a preexisting the at least one branding action of the at least one writable metadata as part of the at least one digital media asset; or to establish permission or denial to execute the at least one branding action or the post read and write requests of the at least one writable metadata. To do this, controlled by the at least one excelsior enabler. The App may establish at least one connection, which is usually through the Internet, with a programmable communications console, which is usually a combination of an API protocol and graphic user interface (GUI) as part of a web service. In addition, the at least one excelsior enabler provides reestablished credentials to the programmable communications console as part of the at least one membership, in which The App is facilitating and monitoring, to authenticate the data communications session used to send and receive data requests between the at least one membership and The App.

In accordance with another embodiment of the present invention, the present invention teaches a method for monitoring access to an encrypted digital media and facilitating unlimited interoperability between a plurality of data processing devices. The method comprises receiving a branding request from at least one communications console of the plurality of data processing devices, the branding request being a read and write request of metadata of the encrypted digital media, the request comprising a membership verification token corresponding to the encrypted digital media. Subsequently, the membership verification token is authenticated, the authentication being performed in connection with a token database. Thereafter, connection with the at least one communications console is established. Afterwards, at least one electronic identification reference is requested from the at

4

least one communications console. Further, the at least one electronic identification reference is received from the at least one communications console. Finally, branding metadata of the encrypted digital media is performed by writing the membership verification token and the electronic identification reference into the metadata.

The present invention is particularly useful for giving users the freedom to use products outside of the device in which the product was acquired and extend unlimited interoperability with other compatible devices.

#### BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, the needs satisfied thereby, and the objects, features, and advantages thereof, reference now is made to the following description taken in connection with the accompanying drawings.

FIG. 1 shows a system for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 2 shows a system for authoring an encrypted digital media according to an embodiment of the present invention.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention.

Skilled artisans will appreciate that elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale. For example, the dimensions of some of the elements in the figures may be exaggerated relative to other elements to help to improve understanding of embodiments of the present invention.

#### DETAILED DESCRIPTION OF THE DRAWINGS

Before describing in detail the particular system and method for personalised digital media access system in accordance with an embodiment of the present invention, it should be observed that the present invention resides primarily in combinations of system components related to the device of the present invention.

Accordingly, the system components have been represented where appropriate by conventional symbols in the drawings, showing only those specific details that are pertinent to understanding the present invention so as not to obscure the disclosure with details that will be readily apparent to those of ordinary skill in the art having the benefit of the description herein.

In this document, relational terms such as ‘first’ and ‘second’, and the like may be used solely to distinguish one entity or action from another entity or action without necessarily requiring or implying any actual such relationship or order between such entities or actions. The terms ‘comprises’, ‘comprising’, or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method,

## US 8,887,308 B2

5

article, or apparatus that comprises a list of elements does not include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. An element preceded by 'comprises . . . a' does not, without more constraints, preclude the existence of additional identical elements in the process, method, article, or apparatus that comprises the element.

The present invention is directed at providing infinite access rights of legally acquired at least one encrypted digital media asset to the content acquirer, explained in this document as the excelsior enabler, and optionally to their recognized friends and family, explained in this document as a plurality of secondary enablers. To explain further, the excelsior enabler and secondary enablers defined comprises human beings or computerized mechanisms programmed to process steps of the invention as would normally be done manually by a human being. Additionally, an apparatus used alone or in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods with a connection are needed (herein referred to as The App). To deliver the requirements of the invention, communicative and connected elements comprise: verification, authentication, electronic ID metadata branding, additional technical branding, and cross-referencing. The connection handling the communicative actions of the invention will usually be the Internet and can also be an internal apparatus cooperative. The App can further be defined as a Windows OS, Apple OS, Linux OS, and other operating systems hosting software running on a machine or device with a capable CPU, memory, and data storage. The App can be even further defined as a system on a chip (SOC), embedded silicon, flash memory, programmable circuits, cloud computing and runtimes, and other systems of automated processes.

The digital media assets used in this system are encrypted usually with an AES cipher and decryption keys are usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connection usually an Internet server. As explained earlier, the system we will discuss will work as a front-end to encrypted files as an authorization agent for decrypted access.

FIG. 1 shows a system **100** for monitoring access to an encrypted digital media according to an embodiment of the present invention. The system **100** includes a first recipient module **102**, an authentication module **104**, a connection module **106**, a request module **108**, a second receipt module **110** and a branding module **112**. The first receipt module **102** receives a branding request from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media and includes a membership verification token corresponding to the encrypted digital media. Examples of the encrypted digital media includes, and are not limited to, one or more of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

Subsequently, the authentication module **104** authenticates the membership verification token. The authentication is performed in connection with a token database. Further, the connection module **106** establishes communication with the at least one communication console.

According to an embodiment of the present invention, the connection is established through one of internet, intranet, Bluetooth, VPN, Infrared and LAN.

According to another embodiment of the present invention, the communication console is a combination of an Application Programmable interface (API) protocol and graphic user

6

interface (GUI) as a part of web service. The API is a set of routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services. The API is either one of language dependent or language independent.

The request module **108** requests at least one electronic identification reference from the at least one communication console. The second receipt module **110** receives the at least one electronic identification reference from the least one communication console. The branding module **112** brands metadata of the encrypted digital media by writing the membership verification token and the electronic identification into the metadata.

FIG. 2 shows a system **200** for authoring an encrypted digital media according to an embodiment of the present invention. The figure includes a selection module **202**, a password module **204**, a customization module **206**, a database module **208** and an encryption module **210**. The selection module **202** facilitates selection of one or more media items to form the encrypted digital media. Examples of the one or more media items include, and are not limited to, one or more of a video, an audio and a game.

According to an embodiment of the present invention, the one or more media items are one or more of remote URL links and local media files.

The password module **204** prompts the user to enter a master password which provides access to the encrypted digital media. Subsequently, the customization module **206** allows the user to customize the user access panel of the encrypted digital media.

According to an embodiment of the present invention, the customization module **206** facilitates adding one or more of a banner, a logo, an image, an advertisement, a tag line, a header message and textual information to the user access panel of the encrypted digital media.

Further, the database module **208** connects the encrypted digital media to a database of membership verification token required for decrypting the encrypted digital media.

According to an embodiment of the present invention, the membership verification token is a kodekey. The kodekey is a unique serial number assigned to the encrypted digital media.

The encryption module **210** encrypts the one or more media items to create the encrypted digital media.

According to an embodiment of the present invention, the system **200** further includes a watermark module. The watermark module watermarks information on the encrypted digital media, wherein the watermark is displayed during playback of the encrypted digital media.

According to another embodiment of the present invention, the system **200** further includes an access module. The access module allows the user to define access rights. Examples of the access rights include, but are not limited to, purchasing rights, rental rights and membership access rights.

According to yet another embodiment of the present invention, the system **200** further includes a name module. The name module allows the user to name the encrypted digital media.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention. The process is achieved by way of an enabler using an apparatus or otherwise known as an application in which facilitates digital media files. The apparatus interacts with all communicative parts required to fulfill the actions of the invention. The figure shows a Kodekey Graphical User Interface (GUI) **301**, a product metadata **302**, a networking card **303**, internet **304**, **306** and **308**, database **305** and **309** and an API website.com GUI **307**. A user posts a branding request via the Kodekey GUI interface **301**. The



US 8,887,308 B2

7

Kodekey GUI interface **301** is the GUI for entering token. The Kodekey GUI interface **301** prompts the user to enter the token and press the redeem button present on the Kodekey GUI interface **301**. The product metadata **302** is read/writable metadata associated with the digital media to be acquired. The networking card **303** facilitates querying of optional metadata branding process and referenced. The Kodekey GUI interface is connected to the database **305** via the internet **304** through the networking card **303**. The database **305** is the database used to read/write and store the tokens, also referred to as token database. The user is redirected to the APIwebsite.com GUI **307** through the internet **306**. The APIwebsite.com is the GUI to the membership API in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **307** prompts the user to enter a login id and a password to access the digital media which is acquired from the database **309** through the internet **308**. The database **309** is the database connected to the web service membership in which the user's electronic ID is queried from.

Examples of the encrypted digital files include, and are not limited to, a video file, an audio file, container formats, documents, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention. Subsequently, the communicative parts to cross-reference information stored in the metadata of the digital media asset are checked which has been previously handled by the process of FIG. 1. The figure shows an enabler access request **401**, a product metadata **402**, a networking card **403**, an internet **404**, **406** and **408**, a database **405** and **409** and an APIwebsite.com GUI **407**. The enabler access request **401** facilitates the user to make a request for the digital media. The product metadata **402** is read/writable metadata associated with the digital media to be acquired. The networking card **403** facilitates querying of optional metadata branding process and referenced. The database **405** is the database used to read/write and store the tokens. The APIwebsite.com GUI **407** is the GUI in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **407** prompts the user to enter a login id and a password to access the digital media from the database **409** through the internet **408**. The database **409** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory. The figure represents an authorization sequence action in which a machine is authorized to accept a personalized digital media file. The figure includes STR3EM Machine GUI **501** including the connect icon **502**, a load key file icon **503**, a networking card **504**, an internet **505**, **508** and **510**, a database **506** and **511**, a machine memory **507** and a APIwebsite.com GUI **509**. The STR3EM Machine GUI **501** prompts the user to connect or load a key file to authorize the device through the connect icon **502** and the load key file icon **503**. The STR3EM Machine GUI **501** is connected to the networking card **504**. The networking card **504** facilitates querying of optional metadata branding process and referenced. Further, the STR3EM machine GUI **501** is connected to the database **506** via the internet **505**. The database **506** is the database used to read/write and store the tokens. Moreover, STR3EM Machine GUI **501** is connected to the machine memory **507**. The machine memory **507** represents the internal memory of the machine or device so authoriza-

8

tions can be saved for access of the digital media. The APIwebsite.com GUI **509** is connected to the STR3EM machine GUI through the internet **508**. Further, APIwebsite.com GUI **509** is connected to the database **511** through the internet **510**. The APIwebsite.com GUI **509** prompts the user to enter the login id and a password to authorize the access to digital media. The database **511** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention. At step **602**, a branding request is made by a user from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media.

According to an embodiment of the present invention, the request includes a membership verification token corresponding to the encrypted digital media.

Subsequently, the membership verification token is authenticated at step **604**. The authentication is performed in connection with a token database. Further, connection with the at least communication console is established at step **606**. Afterwards, at least one electronic identification reference is requested from the at least one communications console at the step **608**. At step **610**, at least one electronic identification reference is received from the at least one communication console. Finally, metadata of the encrypted digital media is branded by writing the membership verification token and the electronic identification reference into the metadata at the step **612**.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention. At step **702**, one or more media items are selected by the user to form the encrypted digital media. Subsequently, a master password is entered for providing access to the encrypted digital media for editing at step **704**. Afterwards, the user customizes the user panel of the encrypted digital media at step **706**. Further, the encrypted digital media is connected to a database of membership verification tokens required for decrypting the encrypted digital media at the step **708**. Finally, the one or more media items are encrypted to create the encrypted digital media at the step **710**.

According to various embodiments of the present invention, the verification is facilitated by at least one token handled by at least one excelsior enabler. Examples of the token include, and are not limited to, a structured or random password, e-mail address associated with an e-commerce payment system used to make an authorization payment, or other redeemable instruments of trade for access rights of digital media. Examples of e-commerce systems are PayPal, Amazon Payments, and other credit card services.

According to an embodiment of the present invention, an identifier for the digital media is stored in a database with another database of a list of associated tokens for cross-reference identification for verification.

According to an embodiment of the present invention, the database of a list of associated tokens includes Instant Payment Notification (IPN) received from successful financial e-commerce transactions that includes the identifier for the digital media; import of CSV password lists, and manually created reference phrases.

For this discussion, the structured or random password example will be used as reference. The structured or random passwords can be devised in encoded schemes to flag the apparatus of permission type such as: 1) Purchases can start a password sequence with "P" following a random number, so



US 8,887,308 B2

9

further example would be “PSJD42349MFJDF”. 2) Rentals can start or end a password sequence with “R” plus (+) the number of days a rental is allowed, for example “R7” included in “R7SJDHFG58473” flagging a seven day rental. 3) Memberships can start or end a password sequence with “M” plus (+) optionally the length of months valid for example “M11DFJGH34KF” would flag an eleven-month membership period.

According to an embodiment of the present invention, the tokens are stored in a relational database such as MySQL or Oracle. Cloud storage systems such as Amazon’s Web Services Simple Storage Solution, or also known as S3, provides a highly available worldwide replicated infrastructure. In addition to S3, monetization offerings such as DevPay offer developers the opportunity to make money from applications developed to use the services.

The verification will reference to the S3 and DevPay services for example purposes only as many options such as FTP, SimpleDB, solid state storage and others can be used to host the token hosting needed for the verification element of this invention. The token represents permission from the content provider to grant access rights to the excelsior enabler and thereafter the plurality of secondary enablers. To set up the verification the content provider can manually or automatically generate a single or a plurality of structured or random password in which will represent the token. By using public or private access of S3 as part of an apparatus, the content provider can create empty text files giving each the name of the passwords generated. Because S3 is associated with a highly available worldwide infrastructure, to check this password token can be done my simply constructing a HTTP request from the apparatus and triggering follow up actions based on either a 200 HTTP response, which means OK at which point the next action can happen, or a 400 HTTP response which means ERROR at which point the verification process is voided. An additional token can be used to provide a flag to the apparatus that the verification element has been fulfilled for an initial verification token. Creating an alternate version of the first token by appending a reference to the end, for example, does this: “M11DFJGH34KF\_user@str3em.com\_01\_01\_11”. In this example, it is defined that the eleven month authorized membership token was verified by a user@str3em.com on Jan. 1, 2011. By providing a second token, the first token becomes locked to ownership by the excelsior enabler preventing unauthorized users from reusing the first token without providing the authentication associated with the alternative referenced second token. In the interest of providers of the apparatus delivering this invention, this document will teach a method of a HTTP PUT calculation scheme for automatic royalty billing and administration for the token element used in the invention. Amazon’s DevPay allow developers to attach monetary charges to data services of S3 offered as an embedded component of the apparatus. By using the “PUT” requests parameter, tokens generated by the apparatus are monitored, calculated, and charged to clients of the apparatus provider. For example: the default charge measure for DevPay is \$0.05 for every 1000 PUT requests. By changing the amount to \$100 for every 1000 PUT requests, the apparatus provider is paid a \$0.10 royalty for each token created. Content providers using a connected apparatus like DevPay to deliver and manage digital media distribution do not need to have restrictions on the tokens created as with prior art DRM key providers as DevPay is charged on a pay-as-you-need model on a monthly basis. As a novelty to the apparatus provider, if a content provider fails to pay royalties due, the DevPay hosting will automatically deny token access to all

10

related media products in distribution and restore this verification element when royalties are paid in full.

The authentication element of this invention is at least handled first by the at least one excelsior enabler with a connection to a membership. In the present discussion, the connection is equal to the Internet and the membership is equal to a web service. Further, the web service must be available for two way data exchange to complete the authentication process of this invention. Data exchange with a web service is usually facilitated with a programmable communications console, at most times, will be an Applications Programmable Interface (API). An API is a set of routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services in order to support the building of applications. An API may be language-dependent: that is, available only in a particular programming language, using the particular syntax and elements of the programming language to make the API convenient to use in this particular context. Alternatively an API may be language-independent: that is, written in a way that means it can be called from several programming languages (typically an assembly/C-level interface). This is a desired feature for a service-style API that is not bound to a particular process or system and is available as a remote procedure call. A more detailed description of API that can be used for an apparatus can be found in the book, “Professional Web APIs with PHP: eBay, Google, Paypal, Amazon, FedEx plus Web Feeds”, by Paul Reinheimer, Wrox publishers (2006). A program apparatus, scripts, often calls these APIs or sections of code residing on user computerized devices. For example, a web browser running on a user computer, cell phone, or other device can download a section of JavaScript or other code from a web server, and then use this code to in turn interact with the API of a remote Internet server system as desired. A Graphic User Interface (GUI) can be installed for human interaction or processes can be preprogrammed in a programmable script such as PHP, ASP.Net, Java, Ruby on Rails and others. The authentication element of the invention is usually embedded as a process of the apparatus but could be linked dynamically. In this document, the embedded version using a GUI will be used as reference. The web service equipped with the API is usually a well-known membership themed application in which the users must use an authentic identification. Some example includes Facebook in which as a rule, members are required to use their legal name identities. A reference number or name with the Facebook Platform API represents this information. Other verified web services in which real member names are required such as the LinkedIn API and the PayPal API and even others could be used, but for this discussion, Facebook will be used only as an example of how the authentication element of the invention is utilized. The Facebook API system, as well as others, operates based on an access authentication system called from a connected apparatus (which is usually an Internet powered desktop or browser based application) with an API Key, an Application Secret Key and could also include an Application ID. For example, the Facebook API Application Keys required to establish a data exchange session with the connected apparatus might look like:

```
API Key
37a925fc5ee9b4752af981b9a30e9a73gh
Application Secret
f2a2d92ef395cce88eb0261d4b4gsa782
Application ID
51920566446
```

The collective API keys are usually embedded in the source code of the apparatus, or stored on a remote Internet server, and could be included in the encrypted digital media metadata

US 8,887,308 B2

11

and inserted on-the-fly into calls made to the API from the connected apparatus. This allows dynamic API connection of the apparatus using keys issued to individual content providers so in the event of a reprimand of a single the individual content provider by the API provider, the collective the individual content providers and the enablers of the connected apparatus are not affected.

Upon an access request of the digital media, the excelsior enabler interacts with the apparatus, usually software or web application, to enter membership credentials in a GUI front-end connected to the API. The membership credentials are usually comprised of a login element comprising a name, phrase, or e-mail address, and a secret password. The credentials can be generated by the enabler or automatically generated by the web service. Once the enabler authenticates their identity with the membership, the apparatus facilitating the data communication can request relevant information to fulfill the process chain of the invention. For example, Facebook API Platform defines members as ID numbers, so if a member's real name is John Doe, then Facebook API ID (also programmatically known as the FBID) would be 39485678. Once the enabler successfully sign in to the GUI element then the apparatus will query the API for at least one electronic identification reference, in this discussion is the FBID. The FBID is received to the permanent or temporary memory of the apparatus to sustain the branding and cross-referencing requirements of the invention. Additional information can be requested according to membership status or connected "friends" of the enabler. Additional information can be made required for successful authentication and includes: a minimum amount of total friends, a minimum amount of female friends, a minimum amount of male friends, a minimum amount of available pictures, a minimum age limit and other custom rules can be defined by the apparatus. An example of how this would work is a content provider can define a minimum of 32 Facebook friends are required to access an encrypted digital media asset offered for sale or promotion. This is achieved by the apparatus handling a access request in which the enabler has not yet acquired access rights by executing and parsing information returned by the Facebook "Friends.get" API command.

XML return example of the Facebook "Friends.get" API command where a plurality of FBID are returned to the apparatus for parsing additional information as may be required to satisfy successful authentication:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_get_response xmlns="http://api.facebook.com/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/
    http://api.facebook.com/1.0/facebook.xsd" list="true">
  <uid>222333</uid>
  <uid>1240079</uid>
</friends_get_response>
```

---

When authenticating a compatible device or machine which may not have access to a connection for the authentication element, a key file or part of the metadata thereof could be made on another connected compatible device or machine and allow the enabler to execute Friends.get API command to collect and store the complete list of a plurality of FBID to the key file or the metadata thereof. The compatible device or machine which may not have access to a connection for the authentication element with an embedded interaction console, usually a user GUI, can request and load the key file or part of the metadata thereof to save the complete list of a plurality of electronic identification references, in this discus-

12

sion is reference as the FBID, to storage or metadata as part of the compatible device or machine. This step ensures the cross-referencing element requirement of the invention can take place in the event the connection for the authentication element is not present in the compatible device or machine.

Another example is a content provider can allow shared access to friends of the excelsior enabler after a time period, like for example, 90 days. After the 90 day period, when media access is requested using the authentication element by a plurality of secondary enablers, which are usually friends and family of the excelsior enabler, the FBID of the excelsior enabler is cross-referenced with the FBID of the requesting secondary enabler by way of the apparatus ability to execute the Facebook "Friends.areFriends" API command.

XML return example of the Facebook "Friends.areFriends" API command where FBID 2223322 and 2222333 are friends and FBID 1240077 and 1240079 are not friends:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_areFriends_response
  xmlns="http://api.facebook.com/1.0/
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/
    http://api.facebook.com/1.0/facebook.xsd" list="true">
  <friend_info>
    <uid1>222332</uid1><uid2>222333</uid2>
    <are_friends>1</are_friends>
  </friend_info>
  <friend_info>
    <uid1>1240077</uid1><uid2>1240079</uid2>
    <are_friends>0</are_friends>
  </friend_info>
</friends_areFriends_response>
```

---

Such usability can be important to sustain "fair use" rights of consumers of the digital media to emulate usability found with physical media products such as CD and DVD that can be loaned to friends and family after an inception grace period.

Once the information of the verification and authentication elements is acquired, the apparatus handles the next process of writing the information to the digital media metadata and can include additional information gathered from components of The App. Components of The App can include MAC address from a networking card, CRC checksum of an embedded file or circuit, SOC identifier, embedded serial number, OS version, web browser version, and many other identifiable components as part of The App. For this discussion, the MAC address from a networking card as part of The App will be used as reference of a secondary electronic identification reference. In computer networking, a Media Access Control address (MAC address) is a unique identifier assigned to most network adapters or network interface cards (NICs) by the manufacturer for identification, and used in the Media Access Control protocol sub-layer. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number. It may also be known as an Ethernet Hardware Address (EHA), hardware address, adapter address, or physical address. The novelty of embedding the MAC address along with the FBID of the excelsior enabler is to provide a plurality of electronic identification references in which cross-referencing actions can allow more rapid access to be granted with less interaction from an enabler. For example, to retrieve the FBID from Facebook to cross-reference with the FBID stored in the digital media metadata requires the enabler to possibly physically need to enter their login and password credentials to the GUI connected to the apparatus. It may be possible that web browser

US 8,887,308 B2

13

cookies allow automatic Facebook login by storing an active session key, but the session key is not guaranteed to be active at the time of an access request. While the enabler may not have an issue executing another authentication command, several remote operations could exist to control authentication and access requests separately from each other. The apparatus can execute a programmable retrieval command, usually a GET command, to locate and retrieve the MAC address from an attached or connected networking card. After the FBID is acquired, the MAC address is also acquired to write the plurality of electronic identifications to the metadata of the at least one encrypted digital media asset by; obtaining the decryption key to decrypt the encrypted digital media asset which is usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connected source, usually an Internet server with an encrypted HTTPS protocol. A plurality of MAC addresses can be stored along with the FBID of the excelsior enabler to manage access rights across a plurality of devices. To understand metadata and the uses, metadata is defined simply as to “describe other data”. It provides information about certain item’s content. For example, an image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document’s metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document. Web pages often include metadata in the form of Meta tags. Description and keywords Meta tags are commonly used to describe the Web page’s content. Most search engines use this data when adding pages to their search index. In the invention, the FBID and MAC addresses are written to the digital media asset metadata to prepare for the instant or delayed cross-referencing element of the invention. The same process of writing the information to the digital media metadata is true with secondary enablers allowing the same benefits of cross-referencing.

Cross-referencing, the last element of the invention is used to verify access rights of an enabler of a pre or post personalized encrypted digital media asset. Once an enabler executes an action for access request, the apparatus will obtain the decryption key to first seek the MAC address record. If the MAC address is found, then a cross-reference process is executed by comparing the MAC address retrieved from the metadata of the digital media file with the MAC address retrieved from the networking card connected to the apparatus or The App. If the comparison action proves to be true, then access rights are granted to the enabler. If the comparison fails, then the apparatus can either ask the enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the enabler. In this discussion, the apparatus requires the enabler to participate in communication with the authentication element to provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook API. If the comparison action proves to be true, then access rights is granted to the excelsior enabler and the current MAC address of the networking card as part of The App is appended to the metadata of the encrypted digital media asset and access rights is granted to the excelsior enabler. If the FBID cross-reference fails, then the apparatus can either ask the potential secondary enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the potential secondary enabler. In this discussion, the apparatus requires the potential secondary enabler to participate in communication with the authentication element to

14

provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook “Friends.areFriends” API command to determine if the potential secondary enabler identity is true or false. The determination is in accordance to any possible access grace periods set by the content provider of the encrypted digital media asset. If the comparison action proves to be true, then access rights is granted to the secondary enabler and the current MAC address of the networking card as part of The App and the FBID retrieved are appended to the established metadata information of the encrypted digital media asset and access rights can be granted to a plurality of secondary enablers; unlimited interoperability between devices and “fair use” sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments is achieved.

While the present invention has been described in connection with preferred embodiments, it will be understood by those skilled in the art that variations and modifications of the preferred embodiments described above may be made without departing from the scope of the invention. Other embodiments will be apparent to those skilled in the art from a consideration of the specification or from a practice of the invention disclosed herein. It is intended that the specification and the described examples are considered exemplary only, with the true scope of the invention indicated by the following claims.

What is claimed is:

1. A process for transforming a user access request for cloud digital content into a computer readable authorization object, the process for transforming comprising:

- a) receiving an access request for cloud digital content through an apparatus in process with at least one CPU, the access request being a write request to a data store, wherein the data store is at least one of:
  - a memory connected to the at least one CPU;
  - a storage connected to the at least one CPU; and
  - a database connected to the at least one CPU through the Internet; wherein
 the access request further comprises verification data provided by at least one user, wherein the verification data is recognized by the apparatus as a verification token; then
- b) authenticating the verification token of (a) using a database recognized by the apparatus of (a) as a verification token database; then
- c) establishing an API communication between the apparatus of (a) and a database apparatus, the database apparatus being a different database from the verification token database of (b) wherein the API is related to a verified web service, wherein the verified web service is a part of the database apparatus, wherein establishing the API communication requires a credential assigned to the apparatus of (a), wherein the apparatus assigned credential is recognized as a permission to conduct a data exchange session between the apparatus of (a) and the database apparatus to complete the verification process, wherein the data exchange session is also capable of an exchange of query data, wherein the query data comprises at least one verified web service account identifier; then
- d) requesting the query data, from the apparatus of (a), from the API communication data exchange session of (c), wherein the query data request is a request for the at least one verified web service identifier;

then

US 8,887,308 B2

15

e) receiving the query data requested in (d) from the API communication data exchange session of (c); and  
f) creating a computer readable authorization object by writing into the data store of (a) at least one of:  
the received verification data of (a); and  
the received query data of (e); wherein  
the created computer readable authorization object is recognized by the apparatus of (a) as user access rights associated to the cloud digital content, wherein the computer readable authorization  
object is processed by the apparatus of (a) using a cross-referencing action during subsequent user access requests to determine one or more of a user access permission for the cloud digital content.

\* \* \* \* \*

15

16

# EXHIBIT 20

USDC SDNY  
DOCUMENT  
ELECTRONICALLY FILED  
DOC #:  
DATE FILED: 4/24/2020

UNITED STATES DISTRICT COURT  
SOUTHERN DISTRICT OF NEW YORK

-----X  
WILLIAM GRECIA,  
  
Plaintiff,  
  
-against-  
  
BANK OF NEW YORK MELLON  
CORPORATION,  
  
Defendant.  
-----  
WILLIAM GRECIA,  
  
Plaintiff,  
  
-against-  
  
CITIBANK, N.A.,  
  
Defendant.  
-----  
WILLIAM GRECIA,  
  
Plaintiff,  
  
-against-  
  
MORGAN STANLEY SMITH BARNEY LLC,  
  
Defendant.  
-----  
WILLIAM GRECIA,  
  
Plaintiff,  
  
-against-  
  
TIAA, FSB d/b/a TIAA Bank,  
  
Defendant.  
-----

19-CV-2810 (VEC)  
19-CV-2811 (VEC)  
19-CV-2812 (VEC)  
19-CV-2813 (VEC)  
19-CV-3278 (VEC)

AMENDED OPINION  
AND ORDER



-----	:
WILLIAM GRECIA,	:
	:
Plaintiff,	:
	:
-against-	:
	:
SAMSUNG ELECTRONICS AMERICA, INC.,	:
	:
Defendant.	:
-----	X

VALERIE CAPRONI, United States District Judge:

The above-captioned cases all involve claims of patent infringement. Plaintiff William Grecia is the inventor of U.S. Patent No. 8,887,308 (the “’308 Patent”), which relates to digital rights management (“DRM”). Technologies within DRM control and limit user access to digital content. According to Grecia, prior art DRM systems could not authorize access to licensed content across multiple devices, such as a phone, tablet, and computer, but, instead, tethered access rights to a particular device. Grecia’s patent includes one claim (“Claim 1”) that teaches a method for transforming a user’s access request into an authorization object that facilitates access across different devices. Each Defendant—four national banks and Samsung Electronics America, Inc.—offers similar online financial applications that Grecia alleges directly infringe upon his patent.

Defendants have moved to dismiss the respective complaints for failure to state a claim under Federal Rule of Civil Procedure 12(b)(6). They argue, *inter alia*, that Claim 1 is patent-ineligible under 35 U.S.C. § 101 and *Alice Corp. v. CLS Bank Int’l*, 573 U.S. 208, 216 (2014). Because the Court agrees, the motions to dismiss are GRANTED.<sup>1</sup>

---

<sup>1</sup> This Amended Opinion supersedes the Court’s Opinion dated March 13, 2020.

## BACKGROUND<sup>2</sup>

The '308 Patent is titled “Digital Cloud Access (PDMAS Part III)” and teaches a DRM solution for accessing licensed content across multiple devices. DRM refers to access control technologies that prevent undesirable or illegal use of digital media content, such as internet-delivered music and video files. Am. Compl. Ex. A (hereinafter, “Spec.”) at 1:29–34. The patent’s innovation is to “brand” digital content—*i.e.* write information to the content’s metadata—with information about the user’s identity and right to access that content. *Id.* at 3:1–8, 4:3–10. The branded information travels with the digital content so that a user can access the content from different devices. The '308 Patent concludes with one claim teaching “a process for transforming a user access request for cloud digital content into a computer readable authorization object.” *Id.* at 14:31–33.

On September 8, 2018, Judge Sullivan construed several of Claim 1’s terms.<sup>3</sup> Am. Compl. ¶ 7 (citing Order, *Grecia v. MasterCard Int’l Inc.*, No. 15-CV-9059 (S.D.N.Y. Sept. 8,

---

<sup>2</sup> On this motion to dismiss, the Court accepts all factual allegations in the pleadings as true and draws all reasonable inferences in the light most favorable to Plaintiff. *See Gibbons v. Malone*, 703 F.3d 595, 599 (2d Cir. 2013). The Court may also “consider any written instrument attached to the complaint, statements or documents incorporated into the complaint by reference, legally required public disclosure documents filed with the [Securities and Exchange Commission], and documents possessed by or known to the plaintiff upon which it relied in bringing the suit.” *Tongue v. Sanofi*, 816 F.3d 199, 209 (2d Cir. 2016) (quoting *ATSI Commc’ns, Inc. v. Shaar Fund, Ltd.*, 493 F.3d 87, 98 (2d Cir. 2007)).

The parties have submitted five separate sets of pleadings for each case. These pleadings are identical in all material respects. For purposes of this Opinion, the Court refers to the pleadings filed in the first-listed captioned case, *Grecia v. Bank of N.Y. Mellon Corp.*, as representative of the rest. The Court uses the following abbreviations: Amended Complaint (Dkt. 37) as “Am. Compl.”; Defendant’s Memorandum of Law in Support of Its Motion to Dismiss the Amended Complaint (Dkt. 40) as “Mem. of Law Supp. Mot.”; Opposition to Motion to Dismiss by Plaintiff William Grecia (Dkt. 41) as “Pl.’s Opp.”; Defendant’s Reply in Support of Its Motion to Dismiss the Amended Complaint (Dkt. 42) as “Reply.”

<sup>3</sup> Judge Sullivan did not rule on any of the issues presented here.

2018), Dkt. 89 (hereinafter, “*MasterCard*”)). In the discussion below, this Court further elaborates the steps of Claim 1 and incorporates Judge Sullivan’s constructions into those steps.

Grecia has filed five nearly identical lawsuits against Bank of New York Mellon, Citibank, Morgan Stanley, and TIAA Bank (collectively, the “Banks”) and Samsung Electronics America (“Samsung”). Grecia alleges that Claim 1 is directed to patentable subject matter under Section 101. *Id.* ¶ 11. He recites the United States Patent and Trademark Office’s examination of Patent ’308 and three decisions denying petitions for *inter partes* review, in which the patent was challenged on Section 102 and 103 grounds. *See id.* ¶¶ 12–16 & Exs. B–E. Grecia then alleges that Defendants’ various finance and payment applications directly infringe upon the patented Claim 1.

The Banks’ payment applications, in particular, interface with another application, Zelle, to allow users to send and receive money without having to store or process a payee’s bank account information. *Id.* ¶ 17 & Ex. F. The Banks’ applications do so by first authenticating a user’s email address or telephone number and then connecting with Zelle. *Id.* ¶¶ 20–21. The applications then request and receive query data from Zelle. *Id.* ¶ 22. Finally, the applications create and process “a computer readable authorization object” to process financial transactions. *Id.* ¶ 23.

Samsung’s application, Samsung Pay, similarly “transform[s] . . . a user’s credit card account number into a payment token that may be used to make purchases,” according to Grecia. Am. Compl. (19-CV-3278, Dkt. 27) ¶ 17. When a user downloads the Samsung Pay application and seeks to use it to make a purchase, he or she must enter a credit card account number that is used to verify permission to make purchases. *Id.* ¶ 18. The application then authenticates the user’s credit card number and connects with a “Samsung Token Requestor,” from which it

requests and receives a “tokenized card number.” *Id.* ¶¶ 19–21. As a last step, it writes that number to the data store on the user’s device and cross-references it during any subsequent requests to make a purchase using Samsung Pay. *Id.* ¶ 22.

## DISCUSSION

### I. Standard of Review

To survive a motion to dismiss under Rule 12(b)(6), “a complaint must allege sufficient facts, taken as true, to state a plausible claim for relief.” *Johnson v. Priceline.com, Inc.*, 711 F.3d 271, 275 (2d Cir. 2013) (citing *Bell Atl. Corp. v. Twombly*, 550 U.S. 544, 555–56 (2007)). “[A] complaint does not need to contain detailed or elaborate factual allegations, but only allegations sufficient to raise an entitlement to relief above the speculative level.” *Keiler v. Harlequin Enters., Ltd.*, 751 F.3d 64, 70 (2d Cir. 2014) (citation omitted). The Court accepts all factual allegations in the complaint as true and draws all reasonable inferences in the light most favorable to the plaintiff. *See Gibbons v. Malone*, 703 F.3d 595, 599 (2d Cir. 2013). The Court, is not, however, “bound to accept as true a legal conclusion couched as a factual allegation.” *Ashcroft v. Iqbal*, 556 U.S. 662, 678 (2009) (quoting *Twombly*, 550 U.S. at 555).

### II. The Motions to Dismiss Are Not Premature

As a threshold matter, Grecia argues that the motions to dismiss are not ripe for resolution because the parties disagree whether the Court must first hold another claim-construction proceeding. *See* Pl.’s Opp. at 1. Grecia’s position has no merit. It is settled law that “[s]ubject matter eligibility under § 101 may be determined at the Rule 12(b)(6) stage of a case.” *ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 765 (Fed. Cir. 2019) (citing *Aatrix Software, Inc. v. Green Shades Software, Inc.*, 882 F.3d 1121, 1125 (Fed. Cir. 2018)). Although it is error for a district court to decide subject-matter eligibility before addressing a claim

construction dispute, the mere existence of such a dispute is not reason to deny a motion to dismiss. *See MyMail, Ltd. v. ooVoo, LLC*, 934 F.3d 1373, 1380 (Fed. Cir. 2019) (citing *Aatrix*, 882 F.3d at 1125). A district court has at least two other options. It may adopt “the non-moving party’s constructions” or otherwise “resolve the disputes to whatever extent is needed to conduct the § 101 analysis, which may well be less than a full, formal claim construction.” *Aatrix*, 882 F.3d at 1125.

Here, the Court accepts Grecia’s proposed constructions. Although there is a dispute whether additional claim construction *proceedings* are necessary (*i.e.*, whether this Court should rely solely upon the prior constructions in *MasterCard*, *see* Proposed Case Management Plan (19-CV-2813, Dkt. 22-1) at 4), the parties do not dispute the meaning of the claim’s particular terms, let alone dispute terms that bear on whether the claim is directed to patent-eligible subject matter. *See MyMail*, 934 at 1380 (holding that the district court erred by ruling on a Rule 12(c) motion where the parties disputed the construction of a term and the district court did not adopt the non-moving party’s construction). Because the Court adopts Grecia’s alleged constructions, there are “no factual allegations that, taken as true, prevent resolving the eligibility question as a matter of law.” *Aatrix*, 882 F.3d at 1125. The Court now proceeds to resolve that question.

### **III. The ’308 Patent Is Directed to Abstract Subject Matter**

The Supreme Court has set out a two-step inquiry to determine whether a patent claims an abstract idea. *See Alice Corp. v. CLS Bank Int’l*, 573 U.S. 208, 216 (2014); *Mayo Collaborative Servs. v. Prometheus Labs., Inc.*, 566 U.S. 66, 72–73 (2012). First, the court must determine whether the claim at issue is directed to an abstract idea. *Alice*, 566 U.S. at 217. If so, the court must determine whether the claim nonetheless “contains an inventive concept sufficient

to transform the claimed abstract idea into a patent-eligible application.” *Id.* at 221 (quotation omitted).

At step one, courts “look at the focus of the claimed advance over the prior art to determine if the claim’s character as a whole is directed to excluded subject matter.” *Chamberlain Grp., Inc. v. Techtronic Indus. Co.*, 935 F.3d 1341, 1346 (Fed. Cir. 2019) (quoting *Affinity Labs of Tex., LLC v. DIRECTV, LLC*, 838 F.3d 1253, 1257 (Fed. Cir. 2016)). “In cases involving software innovations, this inquiry often turns on whether the claims focus on ‘the specific asserted improvement in computer capabilities or, instead, on a process that qualifies as an abstract idea for which computers are invoked merely as a tool.’” *Ancora Techs., Inc. v. HTC Am., Inc.*, 908 F.3d 1343, 1347 (Fed. Cir. 2018), *as amended* (Nov. 20, 2018) (quoting *Finjan, Inc. v. Blue Coat System, Inc.*, 879 F.3d 1299, 1303 (Fed. Cir. 2018)). Thus, a claimed method involving software is patent eligible if it describes a specific improvement over prior art to computer functionality or recites a means particular to computers that solves a problem in an existing technological process. *See Enfish, LLC v. Microsoft Corp.*, 822 F.3d 1327, 1335 (Fed. Cir. 2016); *Koninklijke KPN N.V. v. Gemalto M2M GmbH*, 942 F.3d 1143, 1149–50 (Fed. Cir. 2019). By contrast, a claim is directed to an abstract idea if it does “not claim a particular way of programming or designing the software to [accomplish the claimed functionality], but instead merely claim[s] the resulting systems.” *Apple, Inc. v. Ameranth, Inc.*, 842 F.3d 1229, 1241 (Fed. Cir. 2016). A claim is also directed to an abstract idea if it is “not directed to a specific improvement in the way computers operate.” *Id.*

At step two, courts search for an inventive concept in the claim. An inventive concept is “some element or combination of elements sufficient to ensure that the claim in practice amounts to ‘significantly more’ than a patent on an ineligible concept.” *DDR Holdings, LLC v.*



*Hotels.com, L.P.*, 773 F.3d 1245, 1255 (Fed. Cir. 2014) (quoting *Alice*, 537 U.S. at 218). The court “consider[s] the elements of each claim both individually and as an ordered combination to determine whether the additional elements transform the nature of the claim into a patent-eligible application.” *Alice*, 537 U.S. at 217 (quotation omitted). When a claim is directed to an abstract idea, it must include “additional features to ensure that the claim is more than a drafting effort designed to monopolize the abstract idea.” *Id.* at 221 (quotation omitted). The claim must “involve more than performance of well-understood, routine, and conventional activities previously known to the industry.” *Aatrix*, 882 F.3d at 1128 (quotation omitted). A claim that contains an inventive concept survives an eligibility challenge under Rule 12(b)(6). *Id.* at 1126–27.

Under both steps of *Alice/Mayo*, Claim 1 of Patent ’308 fails.

#### **A. Step One**

The Court finds that Claim 1 is directed to the abstract idea of storing information about permission and identity for processing access requests. It does not teach how to use those two generic pieces of data to process access requests. Nor does it teach any improvement in computer functionality resulting from its method. Instead, Claim 1 risks covering any computer-implemented means of processing an access request by storing information reflecting (i) a user’s permission to access digital content and (ii) the user’s identity.

The ’308 Patent’s specification itself suggests that Claim 1 is directed to an abstract solution for a general problem. According to the specification, prior art DRM technologies have tied authorization to access content to a particular device by writing the device’s ID to the content’s metadata. Spec. at 2:2–9. To validate access rights, those technologies cross-reference metadata to a clearinghouse according to pre-set rules. *Id.* Thus, they “rely on content providers

to maintain computer servers to receive and send session authorization keys to client computers with an Internet connection.” *Id.* at 2:55–57. Consequently, those DRM measures “do not offer a way to provide unlimited interoperability between different machines.” *Id.* at 3:1–3. The ’308 Patent asserts that “a solution is needed to give consumers the unlimited interoperability between devices and ‘fair use’ sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments.” *Id.* at 3:3–8. It then purports to provide that solution: “[a]n object of the present invention is to provide unlimited interoperability of digital media between unlimited machines with management of end-user access to the digital media.” *Id.* at 3:12–14. Although identifying a problem is a start, the specification does not offer a concrete and tangible means for solving the identified problem beyond reciting generic steps, components, and data, evincing an attempt to preempt most, if not all, solutions for interoperability.

Notwithstanding the specification, the Court must give ultimate weight to the method taught in Claim 1. *See ChargePoint*, 920 F.3d at 766 (the specification, although “helpful in illuminating what a claim is directed to . . . must always yield to the claim language in identifying that focus”); *see also Alice*, 573 U.S. at 216 (“the concern that drives” the judicial exceptions to patentability is “one of preemption”).

Claim 1 instructs a six-step “process for transforming a user access request for cloud digital content into a computer readable authorization object.” *Id.* 14:31-33.

First, an apparatus with a database receives a request to access “cloud digital content.”<sup>4</sup> Am. Compl. ¶ 7. The apparatus also receives a write request from a user that includes a

---

<sup>4</sup> As determined in *Mastercard*, “cloud digital content” is just “data capable of being processed by a computer.” *Mastercard* at 11.

verification token (step a). The verification token is “data that represents permission to access digital media or cloud digital content.” *Id.* (quoting *MasterCard* at 15). The apparatus then authenticates the user’s token via a token database (step b).

Next, the apparatus connects through an Application Programmable Interface (“API”) to a second apparatus, comprised of a database and a verified web service (step c). To make that connection, the first apparatus provides a credential that the web service recognizes as permission to exchange data.<sup>5</sup> The first apparatus requests and receives query data from the second apparatus that includes an identifier (steps d and e). Finally, the first apparatus writes the verification data and identifier to its data store, which it can subsequently recognize as access rights to cloud digital content (step f).

Claim 1 reads in full as follows:

- a) receiving an access request for cloud digital content through an apparatus in process with at least one CPU, the access request being a write request to a data store, wherein the data store is at least one of: a memory connected to the at least one CPU; a storage connected to the at least one CPU; and a database connected to the at least one CPU through the Internet; wherein the access request further comprises verification data provided by at least one user, wherein the verification data is recognized by the apparatus as a verification token; then
- b) authenticating the verification token of (a) using a database recognized by the apparatus of (a) as a verification token database; then
- c) establishing an API communication between the apparatus of (a) and a database apparatus, the database apparatus being a different database from the verification token database of (b) wherein the API is related to a verified web service, wherein the verified web service is a part of the database apparatus, wherein establishing the API communication requires a credential assigned to the apparatus of (a), wherein the apparatus assigned credential is recognized as a permission to conduct a data exchange session between the apparatus of (a) and the database apparatus to complete the verification process, wherein the data exchange session is also capable

---

<sup>5</sup> A verified web service means “a web service that is used to authenticate the identity of a user or device.” Am. Compl. ¶ 7 (quoting *Mastercard* at 12).

of an exchange of query data, wherein the query data comprises at least one verified web service account identifier; then

d) requesting the query data, from the apparatus of (a), from the API communication data exchange session of (c), wherein the query data request is a request for the at least one verified web service identifier; then

e) receiving the query data requested in (d) from the API communication data exchange session of (c); and

f) creating a computer readable authorization object by writing into the data store of (a) at least one of: the received verification data of (a); and the received query data of (e); wherein the created computer readable authorization object is recognized by the apparatus of (a) as user access rights associated to the cloud digital content, wherein the computer readable authorization object is processed by the apparatus of (a) using a cross-referencing action during subsequent user access requests to determine one or more of a user access permission for the cloud digital content.

Spec. at 14:34–15:14.

To discern Claim 1’s focus, it is useful “to compare claims at issue to those claims already found to be directed to an abstract idea in previous cases.” *Enfish*, 822 F.3d at 1334. Although Claim 1 claims an improved result—interoperable access to digital content across devices—it does not teach a “specific way to improve the functionality of a computer.” *Koninklijke*, 942 F.3d at 1152. Claim 1’s method amounts to storing proof of permission and proof of identity to a data store, full stop. It teaches that an apparatus, App 1, receives a request to access digital content, which includes the user’s permission to access that content. App 1 then authenticates that the user has authority to access the data and requests proof of the user’s identity from App 2. The method concludes when App 1 writes both pieces of information to its database for cross-reference during subsequent sessions.<sup>6</sup> Claim 1 provides no technical

---

<sup>6</sup> Grecia summarizes Claim 1 similarly, albeit with slightly more specific language that he asserts limits his claim: “The ’308 patent’s solution—fully disclosed and taught in claim 1—is to write both the user’s permission to access the digital content and *the user’s membership account information* into a data store to be cross referenced

explanation for how to enable interoperable access beyond storing and referencing two generic types of information. The computer components serve as a mere “conduit.” *In re TLI Commc’ns LLC Patent Litig.*, 823 F.3d 607, 612 (Fed. Cir. 2016).

The differences between the eligible and ineligible patent claims in *Data Engine Technologies LLC v. Google LLC* are instructive. 906 F.3d 999 (Fed. Cir. 2018). The *Data Engine* patent identified a problem with computer spreadsheets—they “provided little or no tools for creating and managing [what-if] scenarios” to test the extremes of assumptions in a spreadsheet model. *Id.* at 1005 (quotation omitted). One of the patent’s claims recited “specific steps detailing the method of navigating through spreadsheet pages within a three-dimensional spreadsheet environment using notebook tabs.” *Id.* at 1008. That claim required displaying a row of spreadsheet page identifiers in the form of notebook tabs along one side of the first spreadsheet page. *Id.* It also required “at least one user-settable identifying character to label the notebook tab” and described “navigating through the various spreadsheet pages through selection of the notebook tabs.” *Id.* The Federal Circuit found that “the notebook tabs are specific structures within the three-dimensional spreadsheet environment that allow a user to avoid the burdensome task of navigating through spreadsheets in separate windows using arbitrary commands.” *Id.* at 1011. Consequently, the court held that the claim was not directed to an abstract idea but rather to “a specific method for navigating through three-dimensional electronic spreadsheets.” *Id.* at 1008.

---

upon subsequent access requests.” Pl.’s Opp. at 3 (emphasis added). Defendants dispute that Grecia’s characterization accurately reflects Claim 1, as Claim 1 does not include the concept of storing “membership account information.” See Reply at 2–3. Neither party is correct. Contrary to Defendants’ position, the claim language references the concept of a “verified web service,” which Judge Sullivan construed as “a web service that is used to authenticate the identity of a user or device.” Am. Compl. ¶ 7 (quoting *Mastercard* at 12). But that concept is still far more generic than Grecia’s characterization. This dispute, however, is largely academic. Whether the Court construes the term as either party defines it does not affect the Court’s decision.

Another claim in the *Data Engine* patent, by contrast, contained only generic recitations of a method that runs on a computer; not “the specific implementation of a notebook tab interface.” *Id.* at 1012. The claim was comprised of four steps, including: “partitioning [a] plurality of cells into a plurality of two-dimensional cell matrices so that each of the two-dimensional cell matrices can be presented to a user as a spreadsheet page”; “associating each of the cell matrices with a user-settable page identifier”; “creating in a first cell of a first page at least one formula referencing a second cell of a second page said formula including the user-settable page identifier for the second page”; and “storing said first and second pages of the plurality of cell matrices such that they appear to the user as being stored within a single file.” *Id.* at 1011–12. The claim was “not limited to the specific technical solution and improvement in electronic spreadsheet functionality” that made the other claim eligible. *Id.* at 1012. Instead, it covered “any means for identifying electronic spreadsheet pages.” *Id.*

Claim 1 is far more similar to the ineligible *Data Engine* claim than the eligible claim. Like the ineligible claim, Claim 1 is not limited to a specific technical solution in digital rights management. It covers any means for validating a user’s access rights using two generic pieces of information. The *sine qua non* of Claim 1 is two types of information—proof of permission and proof of identity—being stored in a generic data store. And unlike the eligible claim in *Data Engine* (a comparison on which Grecia relies), Claim 1 does not recite any particular improvements in computer technology. To the contrary, Claim 1 shuns any reference to specific computer components, objects, or processes, leaning instead on abstractions like “receiving,” “authenticating,” “data store,” “verification token,” “authorization object,” “CPU,” “API,” and “data exchange session.”



Patent '308's specification does little to rein in the broad preemption proposed by Claim 1. For one, the specification is results-oriented and does not “suggest that the invention involved overcoming some sort of technical difficulty.”<sup>7</sup> *ChargePoint*, 920 F.3d at 768. According to the specification, “the present invention teaches a more personal system of digital rights management which employs electronic ID, as part of a web service membership, to manage access rights across a plurality of devices.” Spec. at 1:24–27. And “[a]n object of the present invention is to provide unlimited interoperability of digital media between unlimited machines with management of end-user access to the digital media.” Spec. at 3:12–14. But the specification does not describe how to implement interoperable access with specific computer components. Although the specification outlines several examples instantiating the method of Claim 1, those examples contain only generic computer parts that carry out routine steps (receiving, authenticating, establishing a communication, requesting, and creating). Claim 1 thus “fails to provide any technical details for the tangible components, but instead predominately describes the [method] in purely functional terms.” *TLI Commc'ns*, 823 F.3d at 612.

Defendants press an analogy to a hall monitor and hall pass which, although not dispositive,<sup>8</sup> illustrates Claim 1's shortcomings. *See* Mem. of Law Supp. Mot. at 3–4, 8. A related analogy that is somewhat closer would be a military police officer (“MP”) dealing with authorization to access a secure military base. The following table summarizes that analogy:

---

<sup>7</sup> Grecia, in his opposition brief, also articulates a solution-oriented focus for Claim 1, reinforcing the Court's conclusion: “Claim 1 is directed to the solution of free, safe access to cloud digital content, requiring that the user's membership and permission to access the digital content be written to a cloud-based data store.” Pl.'s Opp. at 16.

<sup>8</sup> In *Data Engine*, the Federal Circuit reversed the district court for using a real-world analogy to a notebook to find that a patent was directed to an abstract idea. 906 F.3d at 1011. The aptness of an analogy might be telling, but “[i]t is not enough . . . to merely trace the invention to some real-world analogy. That question is reserved for §§ 102 and 103.” *Id.*

Claim 1	Analogy
(a) an apparatus receives a request to access digital content; the request includes a verification token provided by a user.	An MP at the gate of a secure military facility receives a request from a soldier to enter the facility; the soldier shows a military ID.
(b) the apparatus authenticates the verification token.	The MP authenticates the ID.
(c) the apparatus establishes an API communication with a second apparatus.	The MP calls an operator at headquarters.
(d) the apparatus requests query data including a verified web service account identifier through the communication session.	The MP requests information from the operator whether the soldier has orders for that facility.
(e) the apparatus receives the query data.	The MP receives confirmation from the operator that the soldier has been ordered to that facility and provides an authorization number corresponding to the order.
(f) the apparatus creates an authorization object by writing the verification token and identification.	The MP writes into a log the soldier's name from his ID and his authorization number from the operator.

What this analogy demonstrates is that humans can implement the exact process Claim 1 describes without any reference to or reliance upon computers.<sup>9</sup> Claim 1's limitation is nothing more than an abstract and conventional idea applied to computers.

Even assuming the specification is “full of technical details” referencing computer components, the claim it leads to does not require such details. *ChargePoint*, 920 F.3d at 769. Like the patent in *ChargePoint*, Patent '308 is not “intended to improve those particular components,” and nor does Grecia “view[] the combination of those components as [his] invention.” 920 F.3d at 772. Grecia had the idea of storing authorization tokens in a database for cross-device access and retrieval, but Patent '308 goes no further than that. Similarly, in

---

<sup>9</sup> The Court notes that that the hall pass analogy proposed by Defendants was not 100% applicable because it assumed that the student is both the user and the second apparatus and because a hall pass does not typically represent repeat access. The Court's analogy is closer because the MP must communicate both with the soldier, to get his ID, and with an operator, to confirm that the soldier has orders to report to that facility. Moreover, in the Court's analogy, the MP's log book can be used to allow repeated access to the facility. In short, contrary to Grecia's argument, Claim 1 has absolute parallels in the pre-DRM age. See Pl.'s Opp. at 19.

*Ultramercial*, the claims merely recited the abstract idea of “offering media content in exchange for viewing an advertisement,” along with “routine additional steps such as updating an activity log, requiring a request from the consumer to view the ad, restrictions on public access, and use of the Internet.” *Ultramercial, Inc. v. Hulu, LLC*, 772 F.3d 709, 715–16 (Fed. Cir. 2014); *see also SAP Am., Inc. v. InvestPic, LLC*, 898 F.3d 1161, 1163 (Fed. Cir. 2018) (finding that the claims “fit into the familiar class of claims that do not focus on an improvement in computers as tools, but on certain independently abstract ideas that use computers as tools” (quoting *Elec. Power Grp., LLC v. Alstom S.A.*, 830 F.3d 1350, 1354 (Fed. Cir. 2016))).

The claim here is also similar to the claims held to be ineligible in *Uniloc USA, Inc. v. Amazon.com, Inc.*, 243 S. Supp. 3d 797 (E.D. Tex. 2017), *aff’d*, 733 F. App’x 1026 (Fed. Cir. 2018). The claims in that case were a system and method for managing and implementing a time-adjustable license that controlled access to digital content. *Id.* at 800. The claims recited steps such as “*receiving* a request for authorization to use the digital product on a given device” and “*verifying* that a license data associated with the digital product is valid.” *Id.* at 801 (emphasis added). The *Uniloc* court found that the claims were not directed to specific improvements in the functioning of a computer. *Id.* at 804. The claims were instead directed to the abstract idea of time-adjustable licenses and applied that idea to computers for controlling access to licensed software. *Id.* at 804. Similarly, Claim 1 uses generic computer components to control access to digital content through a series of functionally-oriented steps employing conventional computer components. *See also Joao Bock Transaction Sys., LLC v. Jack Henry & Assocs., Inc.*, 76 F. Supp. 3d 513, 522 (D. Del. 2014), *aff’d*, 803 F.3d 667 (Fed. Cir. 2015) (finding that claims merely applied “a conventional business practice utilized by bankers or financial institutions” to computers).

Grecia relies heavily on the prosecution history during which patent examiners rejected three challenges to the '308 patent under 35 U.S.C. §§ 102 and 103. But “it [is not] enough for subject-matter eligibility that claimed techniques be novel and nonobvious in light of prior art, passing muster under 35 U.S.C. §§ 102 and 103.” *SAP*, 898 F.3d at 1163. Grecia’s reliance on the prosecution history betrays that he could not identify a single case with comparable patent claims found eligible under Section 101.

The improvements over prior art in the cases on which Grecia relies involved specific improvements to a computer’s functionality involving particular components. The claim upheld in *Ancora* taught, for example, storing the record of a license for downloaded software in the BIOS memory, rather than other memory. 908 F.3d at 1345. Likewise, the claims in *Visual Memory LLC v. NVIDIA Corp.*, 867 F.3d 1253 (Fed. Cir. 2017), were “directed to a technological improvement: an enhanced computer memory system.” *Id.* at 1259-60. Those claims focused on a “‘specific asserted improvement in computer capabilities’—the use of programmable operational characteristics that are configurable based on the type of processor—instead of ‘on a process that qualifies as an “abstract idea” for which computers are invoked merely as a tool.’” *Id.* (quoting *Enfish*, 822 F.3d at 1336). And, similarly, in *DDR Holdings*, “the claimed solution [was] necessarily rooted in computer technology in order to overcome a problem specifically arising in the realm of computer networks.” 773 F.3d at 1257; *see also Koninklijke*, 942 F.3d at 1153 (holding that claims were eligible because they recited how a data permutation “is used (*i.e.*, modifying the permutation applied to different data blocks), and this specific implementation is a key insight to enabling prior art error detection systems to catch previously undetectable systematic errors”).

In short, the claims in the cases upon which Grecia relies “were patent-eligible because they were directed to improvements in the way computers and networks carry out their basic functions.” *SAP*, 898 F.3d at 1168 (citations omitted). The combination of Claim 1’s steps, by contrast, recites an abstraction with no concrete form.

## **B. Step Two**

Proceeding to *Alice/Mayo* step two, the Court finds that the ’308 Patent does not contain an inventive concept sufficient to transform the claimed abstract idea into a patent-eligible application. The technological components or processes invoked in Claim 1 are not integral to the method’s applications. Claim 1’s reference to an “authorization object,” for example, is just a “‘computer aided limitation’ to a claim covering an abstract concept,” that, “without more, is insufficient to render the claim patent eligible.” *Dealertrack, Inc. v. Huber*, 674 F.3d 1315, 1333 (Fed. Cir. 2012) (quoting *SiRF Tech., Inc. v. Int’l Trade Comm’n*, 601 F.3d 1319, 1333 (Fed. Cir. 2010)). As the MP-access-to-a-secure-facility analogy illustrates, creating the “authorization object” is nothing more than writing down information about permission and identity.<sup>10</sup>

Grecia is not able to identify an inventive concept apart from the abstract idea to which Claim 1 is directed. According to Grecia, Claim 1’s inventive concept is “writing the verified web service account identifier into the data store.” Pl.’s Opp. at 19–20 (citing Am. Compl. ¶¶ 6–7, 11–27). That concept does not qualify as inventive for one fatal reason: it is no different from Claim 1’s ineligible concept of storing information about permission and identity for processing access requests. *See BSG Tech LLC v. Buyseasons, Inc.*, 899 F.3d 1281, 1290 (Fed. Cir. 2018)

---

<sup>10</sup> Claim 1’s invocation of the “cloud” and the “Internet” also does not supply an inventive concept. Merely appealing to the Internet as a channel for the claimed method does not save a patent from ineligibility at the second step of *Alice*. *See Ultramercial*, 772 F.3d at 715 (“Adding routine additional steps . . . and use of the Internet does not transform an otherwise abstract idea into patent-eligible subject matter.”).

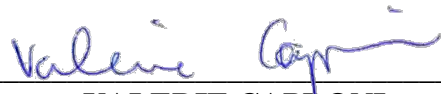
(“It has been clear since *Alice* that a claimed invention’s use of the ineligible concept to which it is directed cannot supply the inventive concept that renders the invention ‘significantly more’ than that ineligible concept.”). Grecia effectively concedes that the “only possible inventive concept in [Claim 1] is the abstract idea itself,” *ChargePoint*, 920 F.3d at 775, when he argues that step two of the analysis is “unnecessary [and] redundant because Claim 1’s solution is also the ‘inventive concept’ required to pass step two.” Pl.’s Opp. at 16. Grecia thus conflates Claim 1’s solution—the abstract idea of storing information about permission and identity for processing access requests—with its purported inventive concept.<sup>11</sup>

### CONCLUSION

For the foregoing reasons, Defendants’ motions are GRANTED. Grecia’s complaints are dismissed with prejudice without leave to amend.

The Clerk of Court is respectfully directed to close all captioned cases and open motions.

**Date: April 24, 2020**  
**New York, New York**

  
\_\_\_\_\_  
**VALERIE CAPRONI**  
**United States District Judge**

---

<sup>11</sup> Because the ’308 Patent’s claim is ineligible, these cases must be dismissed. Accordingly, the Court does not address Defendants’ alternative argument that Grecia has not alleged that their applications perform all the steps of Grecia’s claimed method. *See* Mem. of Law Supp. Mot. at 17.



# EXHIBIT 21



US008533860B1

(12) **United States Patent**  
**Grecia**

(10) **Patent No.:** **US 8,533,860 B1**  
(45) **Date of Patent:** **\*Sep. 10, 2013**

(54) **PERSONALIZED DIGITAL MEDIA ACCESS  
SYSTEM—PDMAS PART II**

(71) Applicant: **William Grecia**, Brooklyn, NY (US)

(72) Inventor: **William Grecia**, Brooklyn, NY (US)

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/740,086**

(22) Filed: **Jan. 11, 2013**

(51) **Int. Cl.**  
**H04L 29/06** (2006.01)

(52) **U.S. Cl.**  
USPC ..... **726/29; 725/28; 713/185; 705/51**

(58) **Field of Classification Search**

None

See application file for complete search history.

2012/0041829 A1	2/2012	Rothschild
2012/0079095 A1	3/2012	Evans
2012/0079126 A1	3/2012	Evans
2012/0079276 A1	3/2012	Evans
2012/0079606 A1	3/2012	Evans
2012/0095871 A1	4/2012	Dorsey
2012/0095906 A1	4/2012	Dorsey
2012/0095916 A1	4/2012	Dorsey
2012/0130903 A1	5/2012	Dorsey
2012/0150727 A1	6/2012	Nuzzi
2012/0166333 A1	6/2012	Von Behren
2012/0173333 A1	7/2012	Berger
2012/0173431 A1	7/2012	Ritchie
2012/0173625 A1	7/2012	Berger
2012/0191553 A1	7/2012	Sathe
2012/0254340 A1	10/2012	Velummylum
2012/0255033 A1	10/2012	Dwivedi
2012/0290376 A1	11/2012	Dryer
2012/0296741 A1	11/2012	Dykes
2012/0310828 A1	12/2012	Hu
2013/0007892 A1	1/2013	Inooka

#### OTHER PUBLICATIONS

Co-pending U.S. Appl. No. 13/397,517 document reference: Jan. 7, 2013 Examiner initiated interview summary (PTOL-413B).

Co-pending U.S. Appl. No. 13/397,517 document reference: Dec. 26, 2012 Advisory Action (PTOL-303).

Co-pending U.S. Appl. 13/397,517 document reference: Nov. 26, 2012 Final Rejection.

(56) **References Cited**

(Continued)

#### U.S. PATENT DOCUMENTS

7,254,235 B2	8/2007	Boudreault et al.	
7,343,014 B2	3/2008	Sovio et al.	
7,526,650 B1 *	4/2009	Wimmer	713/176
8,250,145 B2	8/2012	Zuckerberg	
8,280,959 B1	10/2012	Zuckerberg	
2005/0065891 A1	3/2005	Lee et al.	
2008/0010685 A1	1/2008	Holtzman et al.	
2009/0083541 A1	3/2009	Levine	
2010/0100899 A1	4/2010	Bradbury et al.	
2011/0208695 A1	8/2011	Anand	
2011/0265157 A1	10/2011	Ryder	
2011/0288946 A1 *	11/2011	Baiya et al.	705/26.1
2011/0313898 A1	12/2011	Singhal	
2011/0320345 A1	12/2011	Taveau	

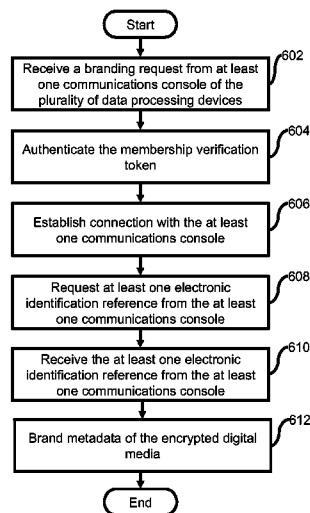
*Primary Examiner* — Jung Kim

*Assistant Examiner* — Tri Tran

(57) **ABSTRACT**

The invention is an apparatus that facilitates access to a data source to accept verification and authentication from an enabler using at least one token and at least one reference. The at least one reference could be a device serial number, a networking MAC address, or a membership ID reference from a web service. Access to the data source is also managed with a plurality of secondary enablers.

**30 Claims, 7 Drawing Sheets**



**US 8,533,860 B1**

Page 2

(56)

**References Cited****OTHER PUBLICATIONS**

Co-pending U.S. Appl. No. 13/397,517 document reference: Nov. 26, 2012.

Co-pending U.S. Appl. No. 13/397,517, document reference: Nov. 26, 2012.

Liu et al. 2004 NPL—A License-sharing scheme in Digital Rights Management.

Co-pending U.S. Appl. No. 13/397,517 document reference: Nov. 26, 2012 Index of Claims.

Co-pending U.S. Appl. No. 13/397,517 document reference: Nov. 26, 2012 Examiner's search strategy and results.

Co-pending U.S. Appl. No. 13/397,517 document reference: Nov. 26, 2012 Non Patent Literature—Baiya et al. U.S. Appl. No. 61/307,196.

Co-pending U.S. Appl. No. 13/397,517 document reference: Nov. 26, 2012 Search information including classification, databases and other search related notes.

Co-pending U.S. Appl. No. 13/397,517 document reference: May 31, 2012 Non-Final Rejection.

Co-pending U.S. Appl. No. 13/397,517 document reference: May 31, 2012.

Co-pending U.S. Appl. No. 13/397,517 document reference: May 31, 2012 Index of Claims.

Co-pending U.S. Appl. No. 13/397,517 document reference: May 31, 2012 Examiner's search strategy and results.

Co-pending U.S. Appl. No. 13/397,517 document reference: May 31, 2012 Bibliographic Data Sheet.

Co-pending U.S. Appl. No. 13/397,517 document reference May 31, 2012 Search information including classification, databases and other search related notes.

Co-pending U.S. Appl. No. 13/397,517 document reference Feb. 4, 2013 Notice of Allowance and Fees Due (PTOL-85).

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Examiner initiated interview summary (PTOL-413B).

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Examiner's Amendment and Detailed Action.

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Issue Information including classification, examiner, name, claim, renumbering, etc.

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013.

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Index of Claims.

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Search information including classification, databases and other search related notes.

Co-pending U.S. Appl. No. 13/397,517 document reference: Feb. 4, 2013 Examiner's search strategy and results.

\* cited by examiner

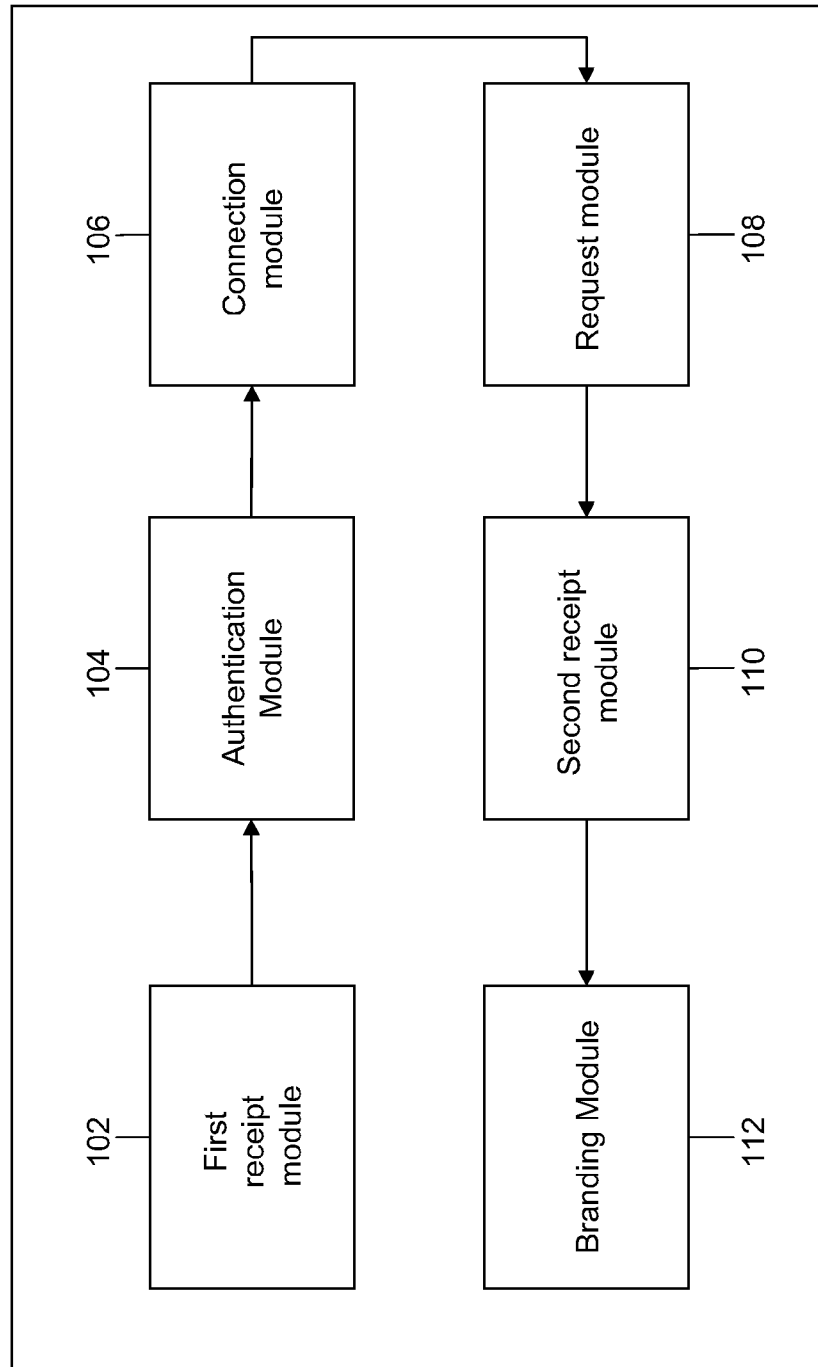


FIG.1

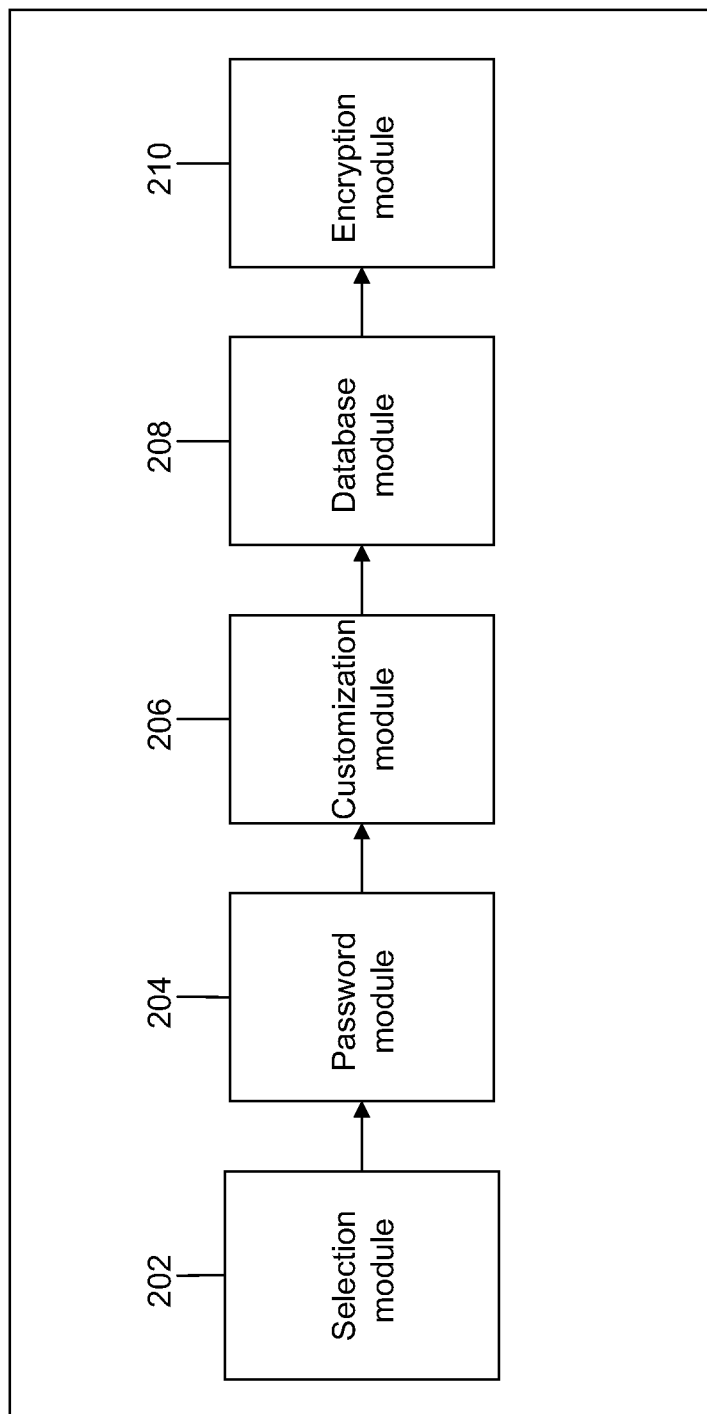


FIG.2

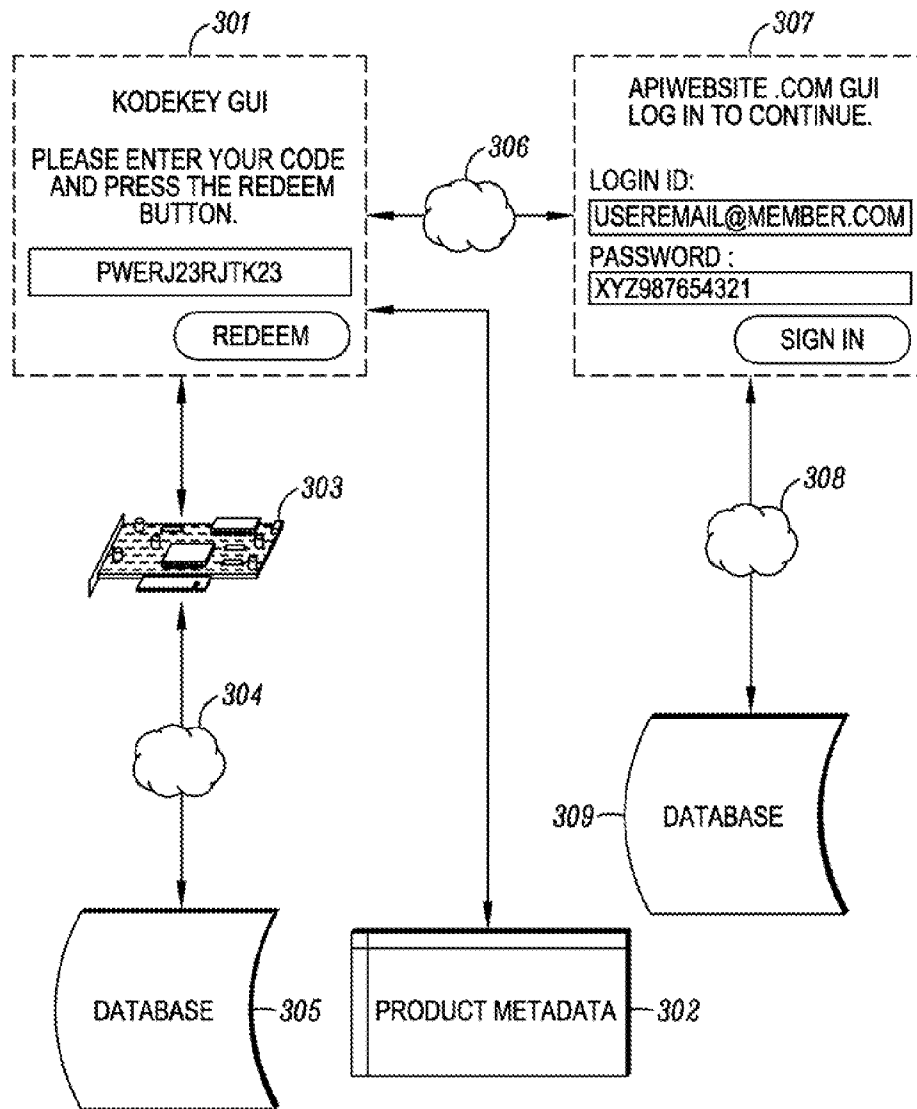


FIG. 3



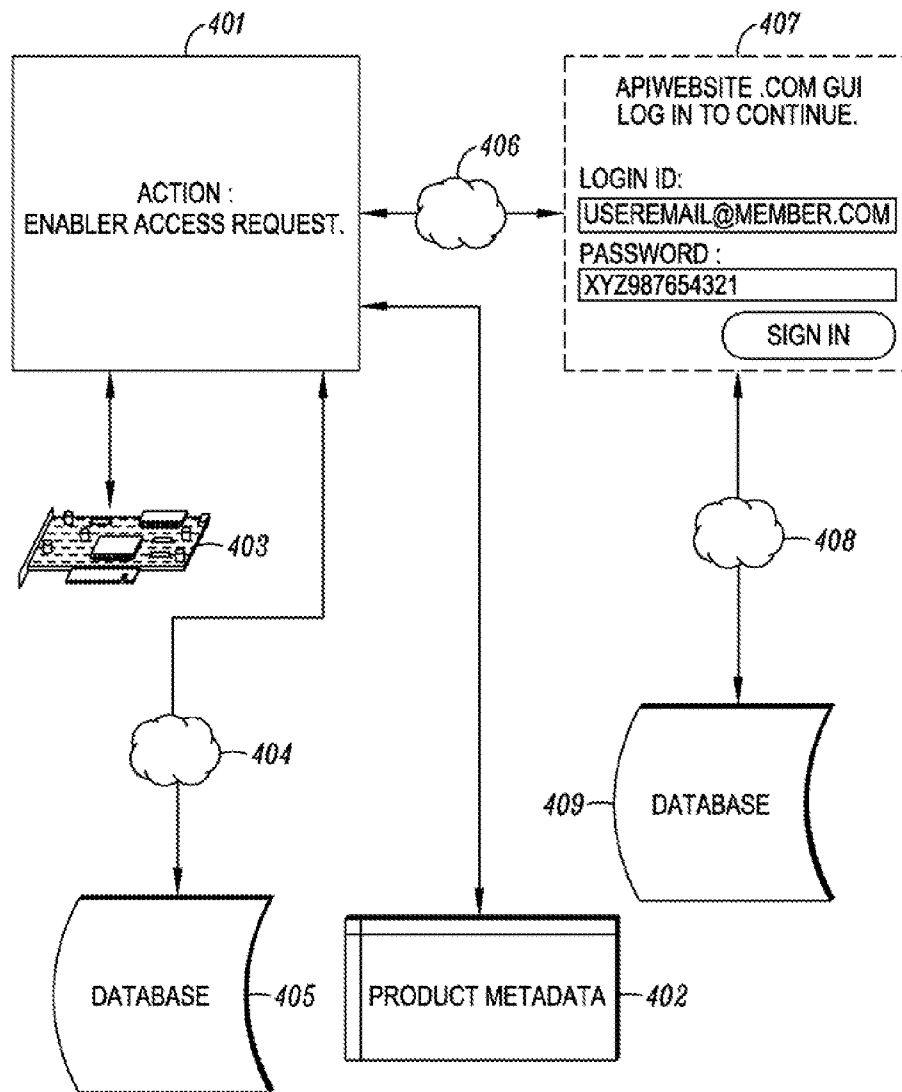


FIG. 4

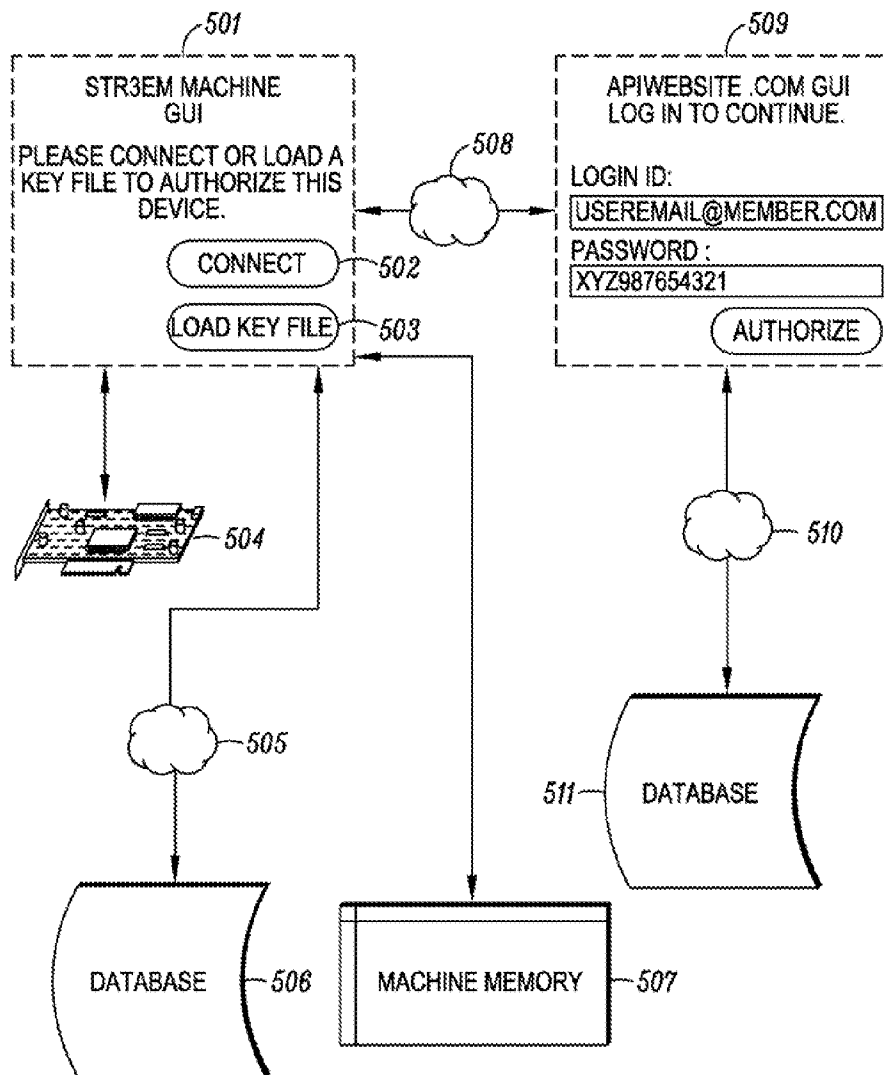


FIG. 5

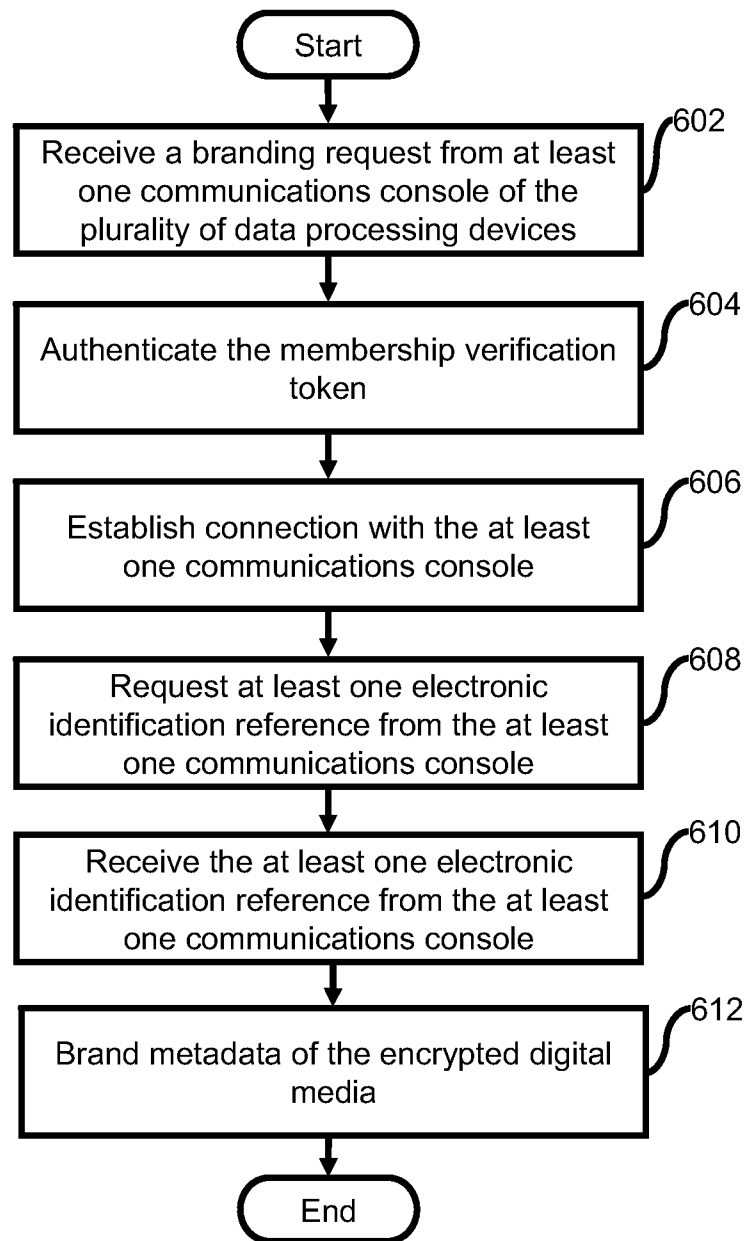


FIG.6

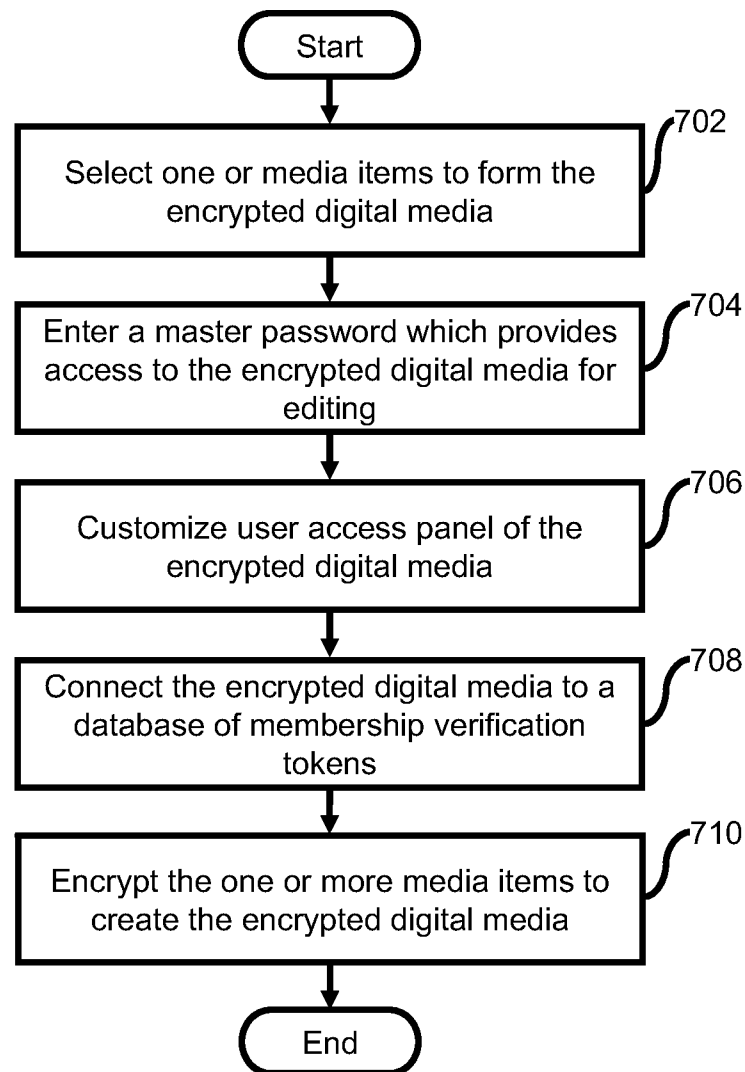


FIG.7

US 8,533,860 B1

1

**PERSONALIZED DIGITAL MEDIA ACCESS  
SYSTEM—PDMAS PART II****CROSS-REFERENCE TO RELATED  
APPLICATIONS**

This application is a continuation of and claims the priority benefit of U.S. patent application Ser. No. 13/397,517 filed Feb. 15, 2012, now pending, which is a continuation of Ser. No. 12/985,351 filed Jan. 6, 2011, now abandoned, which is a continuation of Ser. No. 12/728,218 filed Mar. 21, 2010, now abandoned. Each patent application identified above is incorporated here by reference in its entirety to provide continuity of disclosure.

**BACKGROUND OF THE INVENTION****1. Field of the Invention**

The present invention relates to the field of digital rights management schemes used by creators of electronic products to protect commercial intellectual property copyrights privy to illegal copying using computerized devices. More specifically, the present invention teaches a more personal system of digital rights management which employs electronic ID, as part of a web service membership, to manage access rights across a plurality of devices.

**2. Description of the Prior Art**

Digital rights management (DRM) is a generic term for access control technologies used by hardware manufacturers, publishers, copyright holders and individuals to impose limitations on the usage of digital content across devices. DRM refers to any technology that inhibits undesirable or illegal uses of the digital content. The term generally doesn't refer to forms of copy protection that can be circumvented without modifying the file or device, such as serial numbers or key files. It can also refer to restrictions associated with specific instances of digital works or devices.

Traditional DRM schemes are defined as authentication components added to digital files that have been encrypted from public access. Encryption schemes are not DRM methods but DRM systems are implemented to use an additional layer of authentication in which permission is granted for access to the cipher key required to decrypt files for access. A computer server is established to host decryption keys and to accept authentication keys from Internet connected client computers running client software in which handles the encrypted files. The server can administer different authorization keys back to the client computer that can grant different sets of rules and a time frame granted before the client is required to connect with the server to reauthorize access permissions. In some cases content can terminate access after a set amount of time, or the process can break if the provider of the DRM server ever ceases to offer services.

In the present scenario, consumer entertainment industries are in the transition of delivering products on physical media such as CD and DVD to Internet delivered systems. The Compact Disc, introduced to the public in 1982, was initially designed as a proprietary system offering strict media to player compatibility. As the popularity of home computers and CD-ROM drives rose, so did the availability of CD ripping applications to make local copies of music to be enjoyed without the use of the disc. After a while, users found ways to share digital versions of music in the form of MP3 files that could be easily shared with family and friends over the Internet. The DVD format introduced in 1997 included a new apparatus for optical discs technology with embedded copy protection schemes also recognized as an early form of DRM.

2

With internet delivered music and video files, DRM schemes has been developed to lock acquired media to specific machines and most times limiting playback rights to a single machine or among a limited number of multiple machines regardless of the model number. This was achieved by writing the machine device ID to the metadata of the media file, then cross referencing with a trusted clearinghouse according to pre-set rules. DRM systems employed by DVD and CD technologies consisted of scrambling (also known as encryption) disc sectors in a pattern to which hardware developed to unscramble (also known as decryption) the disc sectors are required for playback. DRM systems built into operating systems such as Microsoft Windows Vista block viewing of media when an unsigned software application is running to prevent unauthorized copying of a media asset during playback. DRM used in computer games such as SecuROM and Steam are used to limit the amount of times a user can install a game on a machine. DRM schemes for e-books include embedding credit card information and other personal information inside the metadata area of a delivered file format and restricting the compatibility of the file with a limited number of reader devices and computer applications.

In a typical DRM system, a product is encrypted using Symmetric block ciphers such as DES and AES to provide high levels of security. Ciphers known as asymmetric or public key/private key systems are used to manage access to encrypted products. In asymmetric systems the key used to encrypt a product is not the same as that used to decrypt it. If a product has been encrypted using one key of a pair it cannot be decrypted even by someone else who has that key. Only the matching key of the pair can be used for decryption. After receiving an authorization token from a first-use action are usually triggers to decrypt block ciphers in most DRM systems. User rights and restrictions are established during this first-use action with the corresponding hosting device of a DRM protected product.

Examples of such prior DRM art include Hurtado (U.S. Pat. No. 6,611,812) who described a digital rights management system, where upon request to access digital content, encryption and decryption keys are exchanged and managed via an authenticity clearing house. Other examples include Alve (U.S. Pat. No. 7,568,111) who teaches a DRM and Tuoriniemi (U.S. Pat. No. 20090164776) who described a management scheme to control access to electronic content by recording use across a plurality of trustworthy devices that has been granted permission to work within the scheme.

Recently, DRM schemes have proven unpopular with consumers and rights organizations that oppose the complications with compatibility across machines manufactured by different companies. Reasons given to DRM opposition range from limited device playback restrictions to the loss of fair-use which defines the freedom to share media products will family members.

Prior art DRM methods rely on content providers to maintain computer servers to receive and send session authorization keys to client computers with an Internet connection. Usually rights are given from the server for an amount of time or amount of access actions before a requirement to reconnect with the server is required for reauthorization. At times, content providers will discontinue servers or even go out of business some years after DRM encrypted content was sold to consumers causing the ability to access files to terminate.

In the light of the foregoing discussion, the current states of DRM measures are not satisfactory because unavoidable issues can arise such as hardware failure or property theft that could lead to a paying customer loosing the right to recover purchased products. The current metadata writable DRM

US 8,533,860 B1

3

measures do not offer a way to provide unlimited interoperability between different machines. Therefore, a solution is needed to give consumers the unlimited interoperability between devices and “fair use” sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments.

#### SUMMARY OF THE INVENTION

An object of the present invention is to provide unlimited interoperability of digital media between unlimited machines with management of end-user access to the digital media.

In accordance with an embodiment of the present invention, the invention is a process of an apparatus which in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods (herein referred to as The App) is used to: handle at least one branding action which could include post read and write requests of at least one writable metadata as part of at least one digital media asset to identify and manage requests from at least one excelsior enabler, and can further identify and manage requests from a plurality of connected second enablers; with at least one token and at least one electronic identification reference received from the at least one excelsior enabler utilizing at least one membership. Here, controlled by the at least one excelsior enabler, The App will proceed to receive the at least one token to verify the authenticity of the branding action and further requests; then establish at least one connection with at least one programmable communications console of the at least one membership to request and receive the at least one electronic identification reference; and could request and receive other data information from the at least one membership. The method then involves sending and receiving variable data information from The App to the at least one membership to verify a preexisting the at least one branding action of the at least one writable metadata as part of the at least one digital media asset; or to establish permission or denial to execute the at least one branding action or the post read and write requests of the at least one writable metadata. To do this, controlled by the at least one excelsior enabler. The App may establish at least one connection, which is usually through the Internet, with a programmable communications console, which is usually a combination of an API protocol and graphic user interface (GUI) as part of a web service. In addition, the at least one excelsior enabler provides reestablished credentials to the programmable communications console as part of the at least one membership, in which The App is facilitating and monitoring, to authenticate the data communications session used to send and receive data requests between the at least one membership and The App.

In accordance with another embodiment of the present invention, the present invention teaches a method for monitoring access to an encrypted digital media and facilitating unlimited interoperability between a plurality of data processing devices. The method comprises receiving a branding request from at least one communications console of the plurality of data processing devices, the branding request being a read and write request of metadata of the encrypted digital media, the request comprising a membership verification token corresponding to the encrypted digital media. Subsequently, the membership verification token is authenticated, the authentication being performed in connection with a token database. Thereafter, connection with the at least one communications console is established. Afterwards, at least one electronic identification reference is requested from the at least one communications console. Further, the at least one

4

electronic identification reference is received from the at least one communications console. Finally, branding metadata of the encrypted digital media is performed by writing the membership verification token and the electronic identification reference into the metadata.

The present invention is particularly useful for giving users the freedom to use products outside of the device in which the product was acquired and extend unlimited interoperability with other compatible devices.

#### BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, the needs satisfied thereby, and the objects, features, and advantages thereof, reference now is made to the following description taken in connection with the accompanying drawings.

FIG. 1 shows a system for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 2 shows a system for authoring an encrypted digital media according to an embodiment of the present invention.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention.

Skilled artisans will appreciate that elements in the figures are illustrated for simplicity and clarity and have not necessarily been drawn to scale. For example, the dimensions of some of the elements in the figures may be exaggerated relative to other elements to help to improve understanding of embodiments of the present invention.

#### DETAILED DESCRIPTION OF THE DRAWINGS

Before describing in detail the particular system and method for personalised digital media access system in accordance with an embodiment of the present invention, it should be observed that the present invention resides primarily in combinations of system components related to the device of the present invention.

Accordingly, the system components have been represented where appropriate by conventional symbols in the drawings, showing only those specific details that are pertinent to understanding the present invention so as not to obscure the disclosure with details that will be readily apparent to those of ordinary skill in the art having the benefit of the description herein.

In this document, relational terms such as ‘first’ and ‘second’, and the like may be used solely to distinguish one entity or action from another entity or action without necessarily requiring or implying any actual such relationship or order between such entities or actions. The terms ‘comprises’, ‘comprising’, or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that comprises a list of elements does not



US 8,533,860 B1

5

include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. An element proceeded by 'comprises . . . a' does not, without more constraints, preclude the existence of additional identical elements in the process, method, article, or apparatus that comprises the element.

The present invention is directed at providing infinite access rights of legally acquired at least one encrypted digital media asset to the content acquirer, explained in this document as the excelsior enabler, and optionally to their recognized friends and family, explained in this document as a plurality of secondary enablers. To explain further, the excelsior enabler and secondary enablers defined comprises human beings or computerized mechanisms programmed to process steps of the invention as would normally be done manually by a human being. Additionally, an apparatus used alone or in accordance with an embodiment, another apparatus, tangible computer medium, or associated methods with a connection are needed (herein referred to as The App). To deliver the requirements of the invention, communicative and connected elements comprise: verification, authentication, electronic ID metadata branding, additional technical branding, and cross-referencing. The connection handling the communicative actions of the invention will usually be the Internet and can also be an internal apparatus cooperative. The App can further be defined as a Windows OS, Apple OS, Linux OS, and other operating systems hosting software running on a machine or device with a capable CPU, memory, and data storage. The App can be even further defined as a system on a chip (SOC), embedded silicon, flash memory, programmable circuits, cloud computing and runtimes, and other systems of automated processes.

The digital media assets used in this system are encrypted usually with an AES cipher and decryption keys are usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connection usually an Internet server. As explained earlier, the system we will discuss will work as a front-end to encrypted files as an authorization agent for decrypted access.

FIG. 1 shows a system **100** for monitoring access to an encrypted digital media according to an embodiment of the present invention. The system **100** includes a first recipient module **102**, an authentication module **104**, a connection module **106**, a request module **108**, a second receipt module **110** and a branding module **112**. The first receipt module **102** receives a branding request from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media and includes a membership verification token corresponding to the encrypted digital media. Examples of the encrypted digital media includes, and are not limited to, one or more of a video file, audio file, container format, document, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

Subsequently, the authentication module **104** authenticates the membership verification token. The authentication is performed in connection with a token database. Further, the connection module **106** establishes communication with the at least one communication console.

According to an embodiment of the present invention, the connection is established through one of internet, intranet, Bluetooth, VPN, Infrared and LAN.

According to another embodiment of the present invention, the communication console is a combination of an Application Programmable interface (API) protocol and graphic user interface (GUI) as a part of web service. The API is a set of

6

routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services. The API is either one of language dependent or language independent.

The request module **108** requests at least one electronic identification reference from the at least one communication console. The second receipt module **110** receives the at least one electronic identification reference from the least one communication console. The branding module **112** brands metadata of the encrypted digital media by writing the membership verification token and the electronic identification into the metadata.

FIG. 2 shows a system **200** for authoring an encrypted digital media according to an embodiment of the present invention. The figure includes a selection module **202**, a password module **204**, a customization module **206**, a database module **208** and an encryption module **210**. The selection module **202** facilitates selection of one or more media items to form the encrypted digital media. Examples of the one or more media items include, and are not limited to, one or more of a video, an audio and a game.

According to an embodiment of the present invention, the one or more media items are one or more of remote URL links and local media files.

The password module **204** prompts the user to enter a master password which provides access to the encrypted digital media. Subsequently, the customization module **206** allows the user to customize the user access panel of the encrypted digital media.

According to an embodiment of the present invention, the customization module **206** facilitates adding one or more of a banner, a logo, an image, an advertisement, a tag line, a header message and textual information to the user access panel of the encrypted digital media.

Further, the database module **208** connects the encrypted digital media to a database of membership verification token required for decrypting the encrypted digital media.

According to an embodiment of the present invention, the membership verification token is a kodekey. The kodekey is a unique serial number assigned to the encrypted digital media.

The encryption module **210** encrypts the one or more media items to create the encrypted digital media.

According to an embodiment of the present invention, the system **200** further includes a watermark module. The watermark module watermarks information on the encrypted digital media, wherein the watermark is displayed during playback of the encrypted digital media.

According to another embodiment of the present invention, the system **200** further includes an access module. The access module allows the user to define access rights. Examples of the access rights include, but are not limited to, purchasing rights, rental rights and membership access rights.

According to yet another embodiment of the present invention, the system **200** further includes a name module. The name module allows the user to name the encrypted digital media.

FIG. 3 shows a flow chart giving an overview of the process of digital media personalization according to an embodiment of the present invention. The process is achieved by way of an enabler using an apparatus or otherwise known as an application in which facilitates digital media files. The apparatus interacts with all communicative parts required to fulfill the actions of the invention. The figure shows a Kodekey Graphical User Interface (GUI) **301**, a product metadata **302**, a networking card **303**, internet **304**, **306** and **308**, database **305** and **309** and an APIwebsite.com GUI **307**. A user posts a branding request via the Kodekey GUI interface **301**. The Kodekey GUI interface **301** is the GUI for entering token. The

US 8,533,860 B1

7

Kodekey GUI interface **301** prompts the user to enter the token and press the redeem button present on the Kodekey GUI interface **301**. The product metadata **302** is read/writable metadata associated with the digital media to be acquired. The networking card **303** facilitates querying of optional metadata branding process and referenced. The Kodekey GUI interface is connected to the database **305** via the internet **304** through the networking card **303**. The database **305** is the database used to read/write and store the tokens, also referred to as token database. The user is redirected to the APIwebsite.com GUI **307** through the internet **306**. The APIwebsite.com is the GUI to the membership API in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **307** prompts the user to enter a login id and a password to access the digital media which is acquired from the database **309** through the internet **308**. The database **309** is the database connected to the web service membership in which the user's electronic ID is queried from.

Examples of the encrypted digital files include, and are not limited to, a video file, an audio file, container formats, documents, metadata as part of video game software and other computer based apparatus in which processed data is facilitated.

FIG. 4 shows a flow chart giving an overview of the process of an access request made by an enabler according to an embodiment of the present invention. Subsequently, the communicative parts to cross-reference information stored in the metadata of the digital media asset are checked which has been previously handled by the process of FIG. 1. The figure shows an enabler access request **401**, a product metadata **402**, a networking card **403**, an internet **404**, **406** and **408**, a database **405** and **409** and an APIwebsite.com GUI **407**. The enabler access request **401** facilitates the user to make a request for the digital media. The product metadata **402** is read/writable metadata associated with the digital media to be acquired. The networking card **403** facilitates querying of optional metadata branding process and referenced. The database **405** is the database used to read/write and store the tokens. The APIwebsite.com GUI **407** is the GUI in which the electronic ID is collected and sent back to the Kodekey GUI interface **301**. The APIwebsite.com GUI **407** prompts the user to enter a login id and a password to access the digital media from the database **409** through the internet **408**. The database **409** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 5 shows personalized digital rights management component as part of a compatible machine with writable static memory. The figure represents an authorization sequence action in which a machine is authorized to accept a personalized digital media file. The figure includes STR3EM Machine GUI **501** including the connect icon **502**, a load key file icon **503**, a networking card **504**, an internet **505**, **508** and **510**, a database **506** and **511**, a machine memory **507** and a APIwebsite.com GUI **509**. The STR3EM Machine GUI **501** prompts the user to connect or load a key file to authorize the device through the connect icon **502** and the load key file icon **503**. The STR3EM Machine GUI **501** is connected to the networking card **504**. The networking card **504** facilitates querying of optional metadata branding process and referenced. Further, the STR3EM machine GUI **501** is connected to the database **506** via the internet **505**. The database **506** is the database used to read/write and store the tokens. Moreover, STR3EM Machine GUI **501** is connected to the machine memory **507**. The machine memory **507** represents the internal memory of the machine or device so authorizations can be saved for access of the digital media. The API-

8

website.com GUI **509** is connected to the STR3EM machine GUI through the internet **508**. Further, APIwebsite.com GUI **509** is connected to the database **511** through the internet **510**. The APIwebsite.com GUI **509** prompts the user to enter the login id and a password to authorize the access to digital media. The database **511** is the database connected to the web service membership in which the user's electronic ID is queried from.

FIG. 6 shows a flowchart for monitoring access to an encrypted digital media according to an embodiment of the present invention. At step **602**, a branding request is made by a user from at least one communications console of the plurality of data processing devices. The branding request is a read and write request of metadata of the encrypted digital media.

According to an embodiment of the present invention, the request includes a membership verification token corresponding to the encrypted digital media.

Subsequently, the membership verification token is authenticated at step **604**. The authentication is performed in connection with a token database. Further, connection with the at least one communication console is established at step **606**. Afterwards, at least one electronic identification reference is requested from the at least one communications console at the step **608**. At step **610**, at least one electronic identification reference is received from the at least one communication console. Finally, metadata of the encrypted digital media is branded by writing the membership verification token and the electronic identification reference into the metadata at the step **612**.

FIG. 7 shows a flowchart showing authoring an encrypted digital media according to an embodiment of the present invention. At step **702**, one or more media items are selected by the user to form the encrypted digital media. Subsequently, a master password is entered for providing access to the encrypted digital media for editing at step **704**. Afterwards, the user customizes the user panel of the encrypted digital media at step **706**. Further, the encrypted digital media is connected to a database of membership verification tokens required for decrypting the encrypted digital media at the step **708**. Finally, the one or more media items are encrypted to create the encrypted digital media at the step **710**.

According to various embodiments of the present invention, the verification is facilitated by at least one token handled by at least one excelsior enabler. Examples of the token include, and are not limited to, a structured or random password, e-mail address associated with an e-commerce payment system used to make an authorization payment, or other redeemable instruments of trade for access rights of digital media. Examples of e-commerce systems are PayPal, Amazon Payments, and other credit card services.

According to an embodiment of the present invention, an identifier for the digital media is stored in a database with another database of a list of associated tokens for cross-reference identification for verification.

According to an embodiment of the present invention, the database of a list of associated tokens includes Instant Payment Notification (IPN) received from successful financial e-commerce transactions that includes the identifier for the digital media; import of CSV password lists, and manually created reference phrases.

For this discussion, the structured or random password example will be used as reference. The structured or random passwords can be devised in encoded schemes to flag the apparatus of permission type such as: 1) Purchases can start a password sequence with "P" following a random number, so further example would be "PSJD42349MFJDF". 2) Rentals

US 8,533,860 B1

9

can start or end a password sequence with “R” plus (+) the number of days a rental is allowed, for example “R7” included in “R7SJDHFG58473” flagging a seven day rental. 3) Memberships can start or end a password sequence with “M” plus (+) optionally the length of months valid for example “M11DFJGH34KF” would flag an eleven-month membership period.

According to an embodiment of the present invention, the tokens are stored in a relational database such as MySQL or Oracle. Cloud storage systems such as Amazon’s Web Services Simple Storage Solution, or also known as S3, provides a highly available worldwide replicated infrastructure. In addition to S3, monetization offerings such as DevPay offer developers the opportunity to make money from applications developed to use the services.

The verification will reference to the S3 and DevPay services for example purposes only as many options such as FTP, SimpleDB, solid state storage and others can be used to host the token hosting needed for the verification element of this invention. The token represents permission from the content provider to grant access rights to the excelsior enabler and thereafter the plurality of secondary enablers. To set up the verification the content provider can manually or automatically generate a single or a plurality of structured or random password in which will represent the token. By using public or private access of S3 as part of an apparatus, the content provider can create empty text files giving each the name of the passwords generated. Because S3 is associated with a highly available worldwide infrastructure, to check this password token can be done my simply constructing a HTTP request from the apparatus and triggering follow up actions based on either a 200 HTTP response, which means OK at which point the next action can happen, or a 400 HTTP response which means ERROR at which point the verification process is voided. An additional token can be used to provide a flag to the apparatus that the verification element has been fulfilled for an initial verification token. Creating an alternate version of the first token by appending a reference to the end, for example, does this: “M11DFJGH34KF\_user@str3em.com\_01\_01\_11”. In this example, it is defined that the eleven month authorized membership token was verified by a user@str3em.com on Jan. 1, 2011. By providing a second token, the first token becomes locked to ownership by the excelsior enabler preventing unauthorized users from reusing the first token without providing the authentication associated with the alternative referenced second token. In the interest of providers of the apparatus delivering this invention, this document will teach a method of a HTTP PUT calculation scheme for automatic royalty billing and administration for the token element used in the invention. Amazon’s DevPay allow developers to attach monetary charges to data services of S3 offered as an embedded component of the apparatus. By using the “PUT” requests parameter, tokens generated by the apparatus are monitored, calculated, and charged to clients of the apparatus provider. For example: the default charge measure for DevPay is \$0.05 for every 1000 PUT requests. By changing the amount to \$100 for every 1000 PUT requests, the apparatus provider is paid a \$0.10 royalty for each token created. Content providers using a connected apparatus like DevPay to deliver and manage digital media distribution do not need to have restrictions on the tokens created as with prior art DRM key providers as DevPay is charged on a pay-as-you-need model on a monthly basis. As a novelty to the apparatus provider, if a content provider fails to pay royalties due, the DevPay hosting will automatically deny token access to all

10

related media products in distribution and restore this verification element when royalties are paid in full.

The authentication element of this invention is at least handled first by the at least one excelsior enabler with a connection to a membership. In the present discussion, the connection is equal to the Internet and the membership is equal to a web service. Further, the web service must be available for two way data exchange to complete the authentication process of this invention. Data exchange with a web service is usually facilitated with a programmable communications console, at most times, will be an Applications Programmable Interface (API). An API is a set of routines, data structures, object classes, and/or protocols provided by libraries and/or operating system services in order to support the building of applications. An API may be language-dependent: that is, available only in a particular programming language, using the particular syntax and elements of the programming language to make the API convenient to use in this particular context. Alternatively an API may be language-independent: that is, written in a way that means it can be called from several programming languages (typically an assembly/C-level interface). This is a desired feature for a service-style API that is not bound to a particular process or system and is available as a remote procedure call. A more detailed description of API that can be used for an apparatus can be found in the book, “Professional Web APIs with PHP: eBay, Google, Paypal, Amazon, FedEx plus Web Feeds”, by Paul Reinheimer, Wrox publishers (2006). A program apparatus, scripts, often calls these APIs or sections of code residing on user computerized devices. For example, a web browser running on a user computer, cell phone, or other device can download a section of JavaScript or other code from a web server, and then use this code to in turn interact with the API of a remote Internet server system as desired. A Graphic User Interface (GUI) can be installed for human interaction or processes can be preprogrammed in a programmable script such as PHP, ASP.Net, Java, Ruby on Rails and others. The authentication element of the invention is usually embedded as a process of the apparatus but could be linked dynamically. In this document, the embedded version using a GUI will be used as reference. The web service equipped with the API is usually a well-known membership themed application in which the users must use an authentic identification. Some example includes Facebook in which as a rule, members are required to use their legal name identities. A reference number or name with the Facebook Platform API represents this information. Other verified web services in which real member names are required such as the LinkedIn API and the PayPal API and even others could be used, but for this discussion, Facebook will be used only as an example of how the authentication element of the invention is utilized. The Facebook API system, as well as others, operates based on an access authentication system called from a connected apparatus (which is usually an Internet powered desktop or browser based application) with an API Key, an Application Secret Key and could also include an Application ID. For example, the Facebook API Application Keys required to establish a data exchange session with the connected apparatus might look like:

```
API Key
37a925fc5ee9b4752af981b9a30e9a73gh
Application Secret
f2a2d92ef395cce88eb0261d4b4gsa782
Application ID
51920566446
```

The collective API keys are usually embedded in the source code of the apparatus, or stored on a remote Internet server, and could be included in the encrypted digital media metadata



US 8,533,860 B1

11

and inserted on-the-fly into calls made to the API from the connected apparatus. This allows dynamic API connection of the apparatus using keys issued to individual content providers so in the event of a reprimand of a single the individual content provider by the API provider, the collective the individual content providers and the enablers of the connected apparatus are not affected.

Upon an access request of the digital media, the excelsior enabler interacts with the apparatus, usually software or web application, to enter membership credentials in a GUI front-end connected to the API. The membership credentials are usually comprised of a login element comprising a name, phrase, or e-mail address, and a secret password. The credentials can be generated by the enabler or automatically generated by the web service. Once the enabler authenticates their identity with the membership, the apparatus facilitating the data communication can request relevant information to fulfill the process chain of the invention. For example, Facebook API Platform defines members as ID numbers, so if a member's real name is John Doe, then Facebook API ID (also programmatically known as the FBID) would be 39485678. Once the enabler successfully sign in to the GUI element then the apparatus will query the API for at least one electronic identification reference, in this discussion is the FBID. The FBID is received to the permanent or temporary memory of the apparatus to sustain the branding and cross-referencing requirements of the invention. Additional information can be requested according to membership status or connected "friends" of the enabler. Additional information can be made required for successful authentication and includes: a minimum amount of total friends, a minimum amount of female friends, a minimum amount of male friends, a minimum amount of available pictures, a minimum age limit and other custom rules can be defined by the apparatus. An example of how this would work is a content provider can define a minimum of 32 Facebook friends are required to access an encrypted digital media asset offered for sale or promotion. This is achieved by the apparatus handling a access request in which the enabler has not yet acquired access rights by executing and parsing information returned by the Facebook "Friends.get" API command.

XML return example of the Facebook "Friends.get" API command where a plurality of FBID are returned to the apparatus for parsing additional information as may be required to satisfy successful authentication:

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_get_response xmlns="http://api.facebook.com/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/ http://api.facebook.com/1.0/facebook.xsd"
  list="true">
  <uid>222333</uid>
  <uid>1240079</uid>
</friends_get_response>
```

When authenticating a compatible device or machine which may not have access to a connection for the authentication element, a key file or part of the metadata thereof could be made on another connected compatible device or machine and allow the enabler to execute Friends.get API command to collect and store the complete list of a plurality of FBID to the key file or the metadata thereof. The compatible device or machine which may not have access to a connection for the authentication element with an embedded interaction console, usually a user GUI, can request and load the key file or part of the metadata thereof to save the complete list of a

12

plurality of electronic identification references, in this discussion is reference as the FBID, to storage or metadata as part of the compatible device or machine. This step ensures the cross-referencing element requirement of the invention can take place in the event the connection for the authentication element is not present in the compatible device or machine.

Another example is a content provider can allow shared access to friends of the excelsior enabler after a time period, like for example, 90 days. After the 90 day period, when media access is requested using the authentication element by a plurality of secondary enablers, which are usually friends and family of the excelsior enabler, the FBID of the excelsior enabler is cross-referenced with the FBID of the requesting secondary enabler by way of the apparatus ability to execute the Facebook "Friends.areFriends" API command.

XML return example of the Facebook "Friends.areFriends" API command where FBID 222332 and 222333 are friends and FBID 1240077 and 1240079 are not friends:

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_areFriends_response
  xmlns="http://api.facebook.com/1.0/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://api.facebook.com/1.0/ http://api.facebook.com/1.0/facebook.xsd"
  list="true">
  <friend_info>
    <uid1>222332</uid1><uid2>222333</uid2>
    <are_friends>1</are_friends>
  </friend_info>
  <friend_info>
    <uid1>1240077</uid1><uid2>1240079</uid2>
    <are_friends>0</are_friends>
  </friend_info>
</friends_areFriends_response>
```

Such usability can be important to sustain "fair use" rights of consumers of the digital media to emulate usability found with physical media products such as CD and DVD that can be loaned to friends and family after an inception grace period.

Once the information of the verification and authentication elements is acquired, the apparatus handles the next process of writing the information to the digital media metadata and can include additional information gathered from components of The App. Components of The App can include MAC address from a networking card, CRC checksum of an embedded file or circuit, SOC identifier, embedded serial number, OS version, web browser version, and many other identifiable components as part of The App. For this discussion, the MAC address from a networking card as part of The App will be used as reference of a secondary electronic identification reference. In computer networking, a Media Access Control address (MAC address) is a unique identifier assigned to most network adapters or network interface cards (NICs) by the manufacturer for identification, and used in the Media Access Control protocol sub-layer. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number. It may also be known as an Ethernet Hardware Address (EHA), hardware address, adapter address, or physical address. The novelty of embedding the MAC address along with the FBID of the excelsior enabler is to provide a plurality of electronic identification references in which cross-referencing actions can allow more rapid access to be granted with less interaction from an enabler. For example, to retrieve the FBID from Facebook to cross-reference with the FBID stored in the digital media metadata requires the enabler to possibly physically need to enter their login and password credentials to the GUI con-

US 8,533,860 B1

13

nected to the apparatus. It may be possible that web browser cookies allow automatic Facebook login by storing an active session key, but the session key is not guaranteed to be active at the time of an access request. While the enabler may not have an issue executing another authentication command, several remote operations could exist to control authentication and access requests separately from each other. The apparatus can execute a programmable retrieval command, usually a GET command, to locate and retrieve the MAC address from an attached or connected networking card. After the FBID is acquired, the MAC address is also acquired to write the plurality of electronic identifications to the metadata of the at least one encrypted digital media asset by; obtaining the decryption key to decrypt the encrypted digital media asset which is usually stored encoded, no encoded, encrypted, or no encrypted as part of the apparatus or as part of a connected source, usually an Internet server with an encrypted HTTPS protocol. A plurality of MAC addresses can be stored along with the FBID of the excelsior enabler to manage access rights across a plurality of devices. To understand metadata and the uses, metadata is defined simply as to "describe other data". It provides information about certain item's content. For example, an image may include metadata that describes how large the picture is, the color depth, the image resolution, when the image was created, and other data. A text document's metadata may contain information about how long the document is, who the author is, when the document was written, and a short summary of the document. Web pages often include metadata in the form of Meta tags. Description and keywords Meta tags are commonly used to describe the Web page's content. Most search engines use this data when adding pages to their search index. In the invention, the FBID and MAC addresses are written to the digital media asset metadata to prepare for the instant or delayed cross-referencing element of the invention. The same process of writing the information to the digital media metadata is true with secondary enablers allowing the same benefits of cross-referencing.

Cross-referencing, the last element of the invention is used to verify access rights of an enabler of a pre or post personalized encrypted digital media asset. Once an enabler executes an action for access request, the apparatus will obtain the decryption key to first seek the MAC address record. If the MAC address is found, then a cross-reference process is executed by comparing the MAC address retrieved from the metadata of the digital media file with the MAC address retrieved from the networking card connected to the apparatus or The App. If the comparison action proves to be true, then access rights are granted to the enabler. If the comparison fails, then the apparatus can either ask the enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the enabler. In this discussion, the apparatus requires the enabler to participate in communication with the authentication element to provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook API. If the comparison action proves to be true, then access rights is granted to the excelsior enabler and the current MAC address of the networking card as part of The App is appended to the metadata of the encrypted digital media asset and access rights is granted to the excelsior enabler. If the FBID cross-reference fails, then the apparatus can either ask the potential secondary enabler to participate in communication with the authentication element of the invention, or could deny further interactivity with the potential secondary enabler. In this discussion, the apparatus requires the potential secondary enabler to par-

14

ticipate in communication with the authentication element to provide credentials to establish a cross-reference comparison with the FBID retrieved from the metadata and the FBID retrieved from the Facebook "Friends.areFriends" API command to determine if the potential secondary enabler identity is true or false. The determination is in accordance to any possible access grace periods set by the content provider of the encrypted digital media asset. If the comparison action proves to be true, then access rights is granted to the secondary enabler and the current MAC address of the networking card as part of The App and the FBID retrieved are appended to the established metadata information of the encrypted digital media asset and access rights can be granted to a plurality of secondary enablers; unlimited interoperability between devices and "fair use" sharing partners for an infinite time frame while protecting commercial digital media from unlicensed distribution to sustain long-term return of investments is achieved.

While the present invention has been described in connection with preferred embodiments, it will be understood by those skilled in the art that variations and modifications of the preferred embodiments described above may be made without departing from the scope of the invention. Other embodiments will be apparent to those skilled in the art from a consideration of the specification or from a practice of the invention disclosed herein. It is intended that the specification and the described examples are considered exemplary only, with the true scope of the invention indicated by the following claims.

What is claimed is:

1. A method for authorizing access to digital content using a cloud system, the cloud system comprising connected modules in operation as one or more of a cloud computing or a cloud storage in connection with devices and users, wherein the digital content is at least one of encrypted or not encrypted, the method facilitating access rights between a plurality of data processing devices, the method comprising:

receiving a digital content access request from at least one communications console of the plurality of data processing devices, the access request being a read or write request of metadata of the digital content, wherein the read or write request of metadata is performed in connection with a combination of at least one device and the cloud system, the request comprising a verification token provided by a first user corresponding to the digital content, wherein the verification token is one or more of a password, e-mail address, payment system, credit card, authorize ready device, rights token, or one or more redeemable instruments of trade;

authenticating the verification token;

establishing a connection with the at least one communications console, wherein the communications console is a combination of a graphic user interface (GUI) and an Application Programmable Interface (API) wherein the API is obtained from a verified web service, the web service capable of facilitating a two way data exchange to complete a verification process wherein the data exchange session comprises at least one identification reference;

requesting the at least one identification reference from the at least one communications console, wherein the identification reference comprises one or more of a verified web service account identifier, letter, number, rights token, e-mail, password, access time, serial number, address, manufacturer identification, checksum, operating system version, browser version, credential, cookie, or key;

US 8,533,860 B1

15

receiving the at least one identification reference from the at least one communications console; and writing at least one of the verification token or the reference into the metadata.

2. The method according to claim 1, wherein the access request being a request from the first user through a data processing device of the plurality of data processing devices; or

wherein the access request being a request from one or more secondary users in network to the first user; wherein the secondary users are validated by a membership web service.

3. The method according to claim 2, wherein the metadata comprises one or more of a software or contents of a web page.

4. The method according to claim 3, wherein the verification token represents verification from a provider of the token to grant access rights to the first user.

5. The method according to claim 3, wherein the digital content is shared among one or more users according to a membership status.

6. The method according to claim 5, wherein the one or more users are a network of recognized human beings using machines or recognized automated computerized mechanisms programmed by human beings, the recognition of the one or more users being validated by the membership status of the membership web service.

7. The method according to claim 6, wherein the digital content access request is from a user using at least one of a computer or a phone hosting an operating system running an application.

8. The method of claim 7, wherein the verification token comprises at least one token selected from the group consisting of a purchase permission, a rental permission, or a membership permission;

wherein the at least one of purchase permission, rental permission, or membership permission is represented by one or more of a letter, number, combination of letters and numbers, rights token, successful payment reference, phrase, name, membership credentials, image, logo, tag, service name, authorization, list, interface button, downloadable program, or an instrument of trade.

9. A system for authorizing access to digital content using a worldwide cloud system infrastructure, the worldwide cloud system infrastructure comprising connected modules in operation as computing and storage, the computing and storage comprising a server, a database, devices and users, wherein the digital content is at least one of encrypted or not encrypted, the system facilitating access rights between a plurality of data processing devices, the system working as a front-end agent for access rights authentication between the plurality of data processing devices, the system further comprising:

a first receipt module, the first receipt module receiving a digital content access request from at least one communications console of the plurality of data processing devices, the access request being a read or write request of metadata of the digital content, wherein the read or write request of metadata is performed in connection with a combination of a device, the server, the database and the cloud system, the metadata further comprises one or more of a software or contents of a web page, the request comprising a verification token provided by a user corresponding to the digital content, wherein the verification token is one or more of a password, e-mail

16

address, payment system, credit card, authorize ready device, rights token, or one or more redeemable instruments of trade;

an authentication module, the authentication module authenticating the verification token;

a connection module, the connection module establishing a connection with the at least one communications console, wherein the communications console is a combination of a graphic user interface (GUI) and an Application Programmable Interface (API) wherein the API is obtained from a verified web service, the web service capable of facilitating a two way data exchange to complete a verification process wherein the data exchange session comprises at least one identification reference;

a request module, the request module requesting the at least one identification reference from the at least one communications console, wherein the identification reference comprises one or more of a verified web service account identifier, letter, number, rights token, e-mail, password, access time, serial number, address, manufacturer identification, checksum, operating system version, browser version, credential, cookie, or key;

a secondary receipt module, the secondary receipt module receiving the at least one identification reference from the at least one communications console; and

a branding module, the branding module writing at least one of the verification token or the identification reference into the metadata.

10. The system of claim 9, wherein the verification token comprises at least one token selected from the group consisting of a purchase permission, a rental permission, or a membership permission;

wherein the at least one of purchase permission, rental permission, or membership permission is represented by one or more of a tag, letter, number, combination of letters and numbers, rights token, successful payment reference, phrase, name, membership credentials, image, logo, service name, authorization, list, interface button, downloadable program, or an instrument of trade.

11. A non-transitory computer medium comprising a program code, the program code being a part of an operating system software or downloaded in sections from a web server, the operating system software program coupled with a user executing a method for authorizing access to digital content wherein the program code, when executed in a processor for facilitating access rights between a plurality of data processing devices, performs the following steps of:

receiving a digital content request from at least one communications console of the plurality of data processing devices, the access request being a read or write request of metadata of the digital content, wherein the read or write request of metadata is performed in connection with a combination of the operating system software program and a cloud system, the request comprising a verification token provided by the user corresponding to the digital content, wherein the verification token is one or more of a password, e-mail address, payment system, credit card, authorize ready device, rights token, key, file, or one or more redeemable instruments of trade;

authenticating the verification token;

establishing a connection with the at least one communications console, wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) wherein the API is obtained from a verified web service, the web service capable of facilitating a two way data exchange



US 8,533,860 B1

17

to complete a verification process wherein the data exchange session comprises at least one identification reference;

requesting the at least one identification reference from the at least one communications console, wherein the identification reference is one or more of a verified web service account identifier, letter, number, rights token, e-mail, password, access time, serial number, address, manufacturer identification, checksum, operating system version, browser version, credential, cookie, or key; receiving the at least one identification reference from the at least one communications console; and writing at least one of the verification token or the identification reference into the metadata.

12. The non-transitory computer medium according to claim 11, wherein the access request is a request from the user providing a credential to a membership web service through a data processing device of the plurality of data processing devices, the user being a human user establishing a permission to the digital content.

13. The non-transitory computer medium of claim 12, wherein the verification token comprises at least one of a purchase permission, a rental permission, or a membership permission;

wherein the at least one of purchase permission, rental permission, or membership permission is represented by one or more of a tag, letter, number, combination of letters and numbers, successful payment reference, phrase, name, membership credential, image, logo, service name, authorization, list, key, file, interface button, downloadable program, or an instrument of trade.

14. The non-transitory computer medium according to claim 13, wherein the verification token represents verification from a provider that a product was acquired.

15. The non-transitory computer medium according to claim 13, wherein the digital content is accessed according to a membership status.

16. The non-transitory computer medium according to claim 15, wherein the membership status is connected to an application programmable interface.

17. The non-transitory computer medium according to claim 15, wherein a remote control operation exist.

18. The non-transitory computer medium according to claim 16, wherein the application programmable interface is connected to a graphic user interface.

19. The non-transitory computer medium according to claim 17, wherein the digital content is shared with one or more secondary users.

20. The non-transitory computer medium according to claim 19, wherein the digital content is shared with the secondary user according to a period of time.

21. A computer product comprising a memory, a CPU, a communications console and a non-transitory computer usable medium, the computer usable medium having an operating system stored therein, the computer product further comprising a customization module, the computer product authorizing access to digital content, wherein the digital content is at least one of an application, a video, or a video game, wherein the digital content is at least one of encrypted or not encrypted, the computer product configured to perform the steps of:

receiving the digital content access request from the communications console, the access request being a read or write request of metadata of the digital content, the metadata of the digital content being one or more of a database or storage in connection to the computer product, the request comprising a verification token corre-

18

sponding to the digital content, the verification token is handled by the user as a redeemable instrument, wherein the verification token comprises at least one of a purchase permission, a rental permission, or a membership permission, wherein the at least one of purchase permission, rental permission, or membership permission being represented by one or more of a tag, a letter, a number, a combination of letters and numbers, a successful payment, a rights token, a phrase, a name, a membership credential, an image, a logo, a service name, an authorization, a list, an interface button, a downloadable program, or the redeemable instrument;

authenticating the verification token;

establishing a connection with the communications console, wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) wherein the API is obtained from a verified web service, the web service capable of facilitating a two way data exchange to complete a verification process wherein the data exchange session comprises at least one identification reference;

requesting the at least one identification reference from the at least one communications console, wherein the identification reference comprises one or more of a verified web service account identifier, letter, number, rights token, e-mail, password, access time, serial number, address, manufacturer identification, checksum, operating system version, browser version, credential, cookie, or key, or ID;

receiving the at least one identification reference from the communications console; and

writing at least one of the verification token or the identification reference into the said metadata.

22. The computer product according to claim 21, wherein the access request is a request from a first user, the first user is a human user in operation of the computer product and establishes first access to the digital content; or

wherein the access request is a request from a secondary user, the secondary user is a human user in operation of the computer product and establishes secondary access to the same digital content as first established for access by the first user.

23. The computer product according to claim 21, wherein the customization module customizes the tag.

24. The computer product according to claim 21, wherein the customization module customizes a user access panel.

25. The computer product according to claim 22, wherein the digital content is monitored for access using a worldwide cloud system infrastructure, the worldwide cloud system infrastructure comprising internet connected modules in operation as computing and storage services in connection to the computer product.

26. The computer product according to claim 22, wherein the verification token is connected to a royalty scheme.

27. The computer product according to claim 25, wherein the access is allowed according to a permission facilitated by a web service working as a front-end agent to the worldwide cloud system infrastructure.

28. The computer product according to claim 25, wherein the computer product is a device of a plurality of devices in a network of the user.

29. The computer product according to claim 28, wherein the device is a computer or a phone.

30. The computer product according to claim 29, wherein a remote control operation exist.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,533,860 B1  
APPLICATION NO. : 13/740086  
DATED : September 10, 2013  
INVENTOR(S) : William Grecia

Page 1 of 1

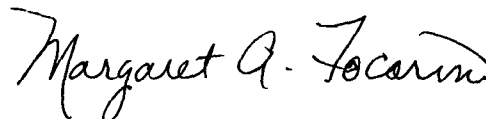
It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims:

In claim 22, column 18, line 37:

“or” should be changed to and read as --and--

Signed and Sealed this  
Seventeenth Day of December, 2013

A handwritten signature in black ink, reading "Margaret A. Focarino". The signature is written in a cursive style with a large initial 'M' and a stylized 'F'.

Margaret A. Focarino  
*Commissioner for Patents of the United States Patent and Trademark Office*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,533,860 B1  
APPLICATION NO. : 13/740086  
DATED : September 10, 2013  
INVENTOR(S) : William Grecia

Page 1 of 2

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On Title page, INSERT:

--(63) Continuation of application No. 13/397,517, filed on Feb. 15, 2012, which is a continuation of application No. 12/985,351, filed on Jan. 6, 2011, which is a continuation of application No. 12/728,218, filed on Mar. 21, 2010, now abandoned.--

In the Claims:

In claim 1, column 15, line 3:

“reference” should be changed to read as --identification reference--

In claim 8, column 15, line 35; and claim 10, column 16, line 31:

“or a membership permission; wherein the at least one of purchase permission, rental permission, or membership permission”,

in each occurrence, should be changed to read as

--and a membership permission; wherein the at least one of purchase permission, rental permission, and membership permission--

In claim 1, column 14, line 56; claim 9, column 16, line 12; claim 11, column 16, line 67; and claim 21, column 18, line 19:

“two way data exchange”, in each occurrence, should be changed to read as --two way data exchange session--

Signed and Sealed this  
Twelfth Day of November, 2013



Teresa Stanek Rea  
Deputy Director of the United States Patent and Trademark Office

**CERTIFICATE OF CORRECTION (continued)**

Page 2 of 2

**U.S. Pat. No. 8,533,860 B1**

In claim 11, column 16, line 65; and claim 21, column 18, line 16 and 17:

“Applications”, in each occurrence, should be changed to --Application--

In claim 1, column 14, line 55; claim 9, column 16, line 11; claim 11, column 16, line 66; and claim 21, column 18, line 18:

“obtained from”, in each occurrence, should be changed to --related to--

In claim 21, column 17, line 62:

“receiving the digital content access request” should be changed to --receiving a digital content access request--

In claim 21, column 18, line 2:

“handled by the user” should be changed to --handled by a user--

# EXHIBIT 22

UNITED STATES DISTRICT COURT  
FOR THE SOUTHERN DISTRICT OF NEW YORK

WILLIAM GRECIA,

Plaintiff,

vs.

SAMSUNG ELECTRONICS AMERICA,  
INC.,

Defendant.

Civil Action No. 1:16-cv-09691-RJS

**STIPULATION AND JOINT MOTION FOR ENTRY OF FINAL JUDGMENT BASED  
ON THE COURT'S INDEFINITENESS RULINGS**

Based on the Court's September 8, 2018 Order, Dkt. No. 58, and as further set forth herein, Plaintiff William Grecia ("Grecia") and Defendant Samsung Electronics America, Inc. ("Samsung"), by and through their respective counsel, hereby stipulate and jointly move the Court for entry of final judgment on Grecia's Claim for Relief for patent infringement as a result of the Court's findings of indefiniteness under 35 U.S.C. § 112, and therefore invalidity, of claims 9, 10, and 21-30 in U.S. Patent No. 8,533,860 (the "'860 Patent").

In light of the foregoing, IT IS HEREBY STIPULATED AND AGREED by the Parties, subject to approval by the Court, as follows:

1. On December 15, 2016, Grecia filed his complaint against Samsung alleging, among other things, that Samsung infringed the '860 Patent. Dkt. No. 1.

2. On February 2, 2017, Samsung answered Grecia's complaint, denying the material allegations and asserting, among other things, defenses of non-infringement and



invalidity of the '860 Patent and a counterclaim for a declaration of noninfringement and invalidity. Dkt. No. 17.

3. On May 27, 2017, Grecia asserted in his Disclosure of Asserted Claims and Infringement Contentions that claims 21, 22, 24, 25, 27, 28, 29, and 30 of the '860 Patent (the "Asserted Claims") were being infringed by Samsung.

4. On September 8, 2018, following briefing and a hearing, the Court issued an Order construing certain terms in the Asserted Claims. Dkt No. 58. The Order found, among other things, that certain terms in the Asserted Claims are indefinite and the Asserted Claims are therefore invalid. Order at 19-25, 27.

5. Based on the Court's September 8, 2018 Order, Dkt. No. 58, Grecia and Samsung stipulate to the Court's entering final judgment of invalidity of the Asserted Claims on the ground that the Court has ruled that certain terms recited in each of the Asserted Claims are indefinite under 35 U.S.C. § 112.

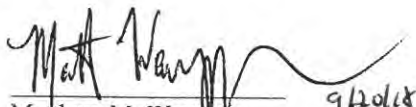
6. Further, because the Order and stipulated final judgment have rendered Samsung's counterclaims moot, Grecia and Samsung further stipulate to the dismissal of all of Samsung's asserted counterclaims without prejudice.

7. By entering into this stipulation and motion, Grecia and Samsung agree that Grecia is not waiving, but rather is expressly reserving, his right to obtain appellate review of the Order by the United States Court of Appeals for the Federal Circuit.


8. Grecia and Samsung reserve their rights to raise on appeal any and all properly appealable issues concerning claim interpretation and/or claim indefiniteness raised by either party in the District Court and/or by the Court's Order.

9. Grecia and Samsung agree that, in the event the United States Court of Appeals for the Federal Circuit modifies or reverses any portion of the Court's Order, or remands back to the Court for further proceedings, nothing in this Stipulation shall restrict in any way a party's ability to re-raise any and all defenses or counterclaims previously raised in this case.

10. For the reasons described herein, Grecia and Samsung jointly and respectfully request that the Court enter a final judgment of invalidity of the Asserted Claims of the '860 Patent.

  
 Matthew M. Wawrzyn  
 matt@wawrzynlaw.com  
 WAWRZYN & JARVIS LLC  
 2700 Patriot Boulevard, Suite 250  
 Glenview, IL 60026  
 Phone: 847.656.5864


*Counsel for Plaintiff William Grecia*

  
 Neil P. Sirota  
 Robert L. Maier  
 Jonathan D. Cocks  
 BAKER BOTTS L.L.P.  
 30 Rockefeller Plaza  
 New York, New York 10112  
 Telephone: (212) 408-2500  
 email: [neil.sirota@bakerbotts.com](mailto:neil.sirota@bakerbotts.com)  
 email: [robert.maier@bakerbotts.com](mailto:robert.maier@bakerbotts.com)  
 email: [jonathan.ocks@bakerbotts.com](mailto:jonathan.ocks@bakerbotts.com)

Timothy S. Durst  
 BAKER BOTTS L.L.P.  
 2001 Ross Avenue  
 Dallas, Texas 75201  
 Telephone: (214) 953-6500  
 email: [tim.durst@bakerbotts.com](mailto:tim.durst@bakerbotts.com)

Attorneys for Defendant/Counter-Plaintiff  
 SAMSUNG ELECTRONICS AMERICA, INC.

IT IS SO ORDERED this 21 day of September 2018.

  
 RICHARD J. SULLIVAN  
 United States District Judge

# EXHIBIT 23

NOTE: This disposition is nonprecedential.

## United States Court of Appeals for the Federal Circuit

---

**WILLIAM GRECIA,**  
*Plaintiff-Appellant*

v.

**SAMSUNG ELECTRONICS AMERICA, INC.,**  
*Defendant-Appellee*

---

2019-1019

---

Appeal from the United States District Court for the  
Southern District of New York in No. 1:16-cv-09691-RJS,  
Judge Richard J. Sullivan.

---

Decided: August 20, 2019

---

MATTHEW MICHAEL WAWRZYN, Wawrzyn & Jarvis LLC,  
Glenview, IL, argued for plaintiff-appellant.

MICHAEL HAWES, Baker Botts, LLP, Houston, TX, ar-  
gued for defendant-appellee. Also represented by TIMOTHY  
S. DURST, Dallas, TX; JONATHAN DREW BRIT, NEIL P.  
SIROTA, New York, NY.

---

Before LOURIE, CHEN, and STOLL, *Circuit Judges*.

CHEN, *Circuit Judge*.

William Grecia asserted claims 21, 22, 24, 25, and 27–30 of U.S. Patent No. 8,533,860 ('860 patent) in a patent infringement suit against Samsung Electronics America, Inc. in *Grecia v. Samsung Electronics America, Inc.*, No. 16–cv–9691 (S.D.N.Y. Sept. 7, 2018). At claim construction, the district court concluded that claim 21 invokes 35 U.S.C. § 112, ¶ 6 and is indefinite under § 112, ¶ 2.<sup>1</sup> Because claims 22, 24, 25, and 27–30 depend from claim 21, the district court ruled that those claims were likewise indefinite. For the reasons below, we *affirm* the district court's decision that claim 21, and thus also dependent claims 22, 24, 25, and 27–30, are invalid as indefinite.

#### BACKGROUND

The '860 patent relates to digital rights management (DRM) technology, which the specification defines as “a generic term for access control technologies used . . . to impose limitations on the usage of digital content across devices.” '860 patent at col. 1, ll. 28–31. According to the specification, the implementation of traditional DRM systems did not permit users to access their digital content on multiple devices or share their content with others. *Id.* at col. 2, l. 67 – col. 3, l. 7. For example, the specification explains that traditional systems rely on content providers to maintain servers for receiving and sending authorization keys, but these content providers might discontinue the servers or go out of business, leaving consumers unable to access the digital content. *Id.* at col. 2, ll. 54–62. The '860 patent purports to solve these problems by storing

---

<sup>1</sup> Paragraphs 2 and 6 of 35 U.S.C. § 112 were replaced by § 112(b) and § 112(f) respectively when the AIA took effect. Because the application resulting in the '860 patent was filed before that date, we refer to the pre-AIA version of § 112.

user authorization information in the digital content's metadata (referred to as "branding" the metadata). *Id.* at col. 3, l. 52 – col. 4, l. 5. The '860 patent explains that access rights to the digital content can then be granted by cross-referencing user information against what is stored in the metadata of that digital content. *Id.* at col. 13, ll. 32–66.

Claim 21, which is the only claim at issue, is reproduced below:

21. A computer product comprising a memory, a CPU, a communications console and a non-transitory computer usable medium, the computer usable medium having an operating system stored therein, the computer product further comprising a *customization module*, the computer product authorizing access to digital content, wherein the digital content is at least one of an application, a video, or a video game, wherein the digital content is at least one of encrypted or not encrypted, the computer product configured to perform the steps of:

receiving a digital content access request from the communications console, the access request being a read or write request of metadata of the digital content, the metadata of the digital content being one or more of a database or storage in connection to the computer product, the request comprising a verification token corresponding to the digital content, the verification token is handled by a user as a redeemable instrument, wherein the verification token comprises at least one of a purchase permission, a rental permission, or a membership permission, wherein the at least one of purchase permission, rental permission, or membership permission being represented by one or more of a tag, a letter, a number, a combination of letters and numbers, a successful payment, a rights token, a



phrase, a name, a membership credential, an image, a logo, a service name, an authorization, a list, an interface button, a downloadable program, or the redeemable instrument;

authenticating the verification token;

establishing a connection with the communications console, wherein the communications console is a combination of a graphic user interface (GUI) and an Applications Programmable Interface (API) wherein the API is obtained from a verified web service, the web service capable of facilitating a two way data exchange session to complete a verification process wherein the data exchange session comprises at least one identification reference;

requesting the at least one identification reference from the at least one communications console, wherein the identification reference comprises one or more of a verified web service account identifier, letter, number, rights token, e-mail, password, access time, serial number, address, manufacturer identification, checksum, operating system version, browser version, credential, cookie, or key, or ID;

receiving the at least one identification reference from the communications console; and

writing at least one of the verification token or the identification reference into the said metadata.

'860 patent at claim 21 (emphasis added).

The computer product of claim 21 includes “a CPU, a communications console, and a non-transitory computer usable medium,” as well as a “customization module.” '860 patent at claim 21. Claim 21 is otherwise silent with regards to the customization module, but the specification explains that the module “allows the user to customize the

user access panel of the encrypted digital media.” *Id.* at col. 6, ll. 26–28. For example, the customization module “facilitates adding one or more of a banner, a logo, an image, an advertisement, a tag line, a header message and textual information to the user access panel of the encrypted digital media.” *Id.* at col. 6, ll. 29–33.

The district court found that the terms “customization module” and “computer product” in claim 21 each invoked § 112, ¶ 6, and were indefinite under § 112, ¶ 2. J.A. 19–24. The district court also found that the term “metadata” was indefinite under § 112, ¶ 2. J.A. 24–25. Grecia appeals the district court’s indefiniteness determinations, focusing its arguments solely on claim 21. This court has jurisdiction over this appeal under 28 U.S.C. § 1295(a)(1).

#### DISCUSSION

##### A. The Claim Term “Customization Module” Invokes § 112, ¶ 6

We review the district court’s claim construction here de novo because it relied only on evidence intrinsic to the ’860 patent. *See Teva Pharm. USA, Inc. v. Sandoz, Inc.*, — U.S. —, 135 S.Ct. 831, 841 (2015). To determine whether § 112, ¶ 6 applies to a claim limitation, the essential inquiry is “whether the words of the claim are understood by persons of ordinary skill in the art to have a sufficiently definite meaning as the name for structure.” *Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1348 (Fed. Cir. 2015). We have traditionally looked to whether the limitation uses the word “means.” If so, there is a rebuttable presumption that § 112, ¶ 6 applies; if not, there is a rebuttable presumption that the provision does not apply. *Id.* at 1348–49. Where, as here, a claim term lacks the word “means,” the presumption can be overcome and § 112, ¶ 6 will apply if the challenger demonstrates that the claim term fails to recite sufficiently definite structure for performing the claimed function. *Id.* at 1349.

As we have explained and the district court correctly noted, “[m]odule’ is a well-known nonce word that can operate as a substitute for ‘means.’” *Williamson*, 792 F.3d at 1350. This is because “[t]he word ‘module’ does not provide any indication of structure because it sets forth the same black box recitation of structure . . . as if the term ‘means’ had been used.” *Id.* Nor does the prefix “customization” impart structure, because it at best describes the module’s intended functionality. And “customization module” stands alone in claim 21—there is nothing in the claim that describes the customization module, let alone imparts structure to the term.

On appeal, Grecia devotes ten pages of his Appeal Brief to the heading “Claim 21’s ‘Computer Product’ or ‘Customization Module’ Terms Do *Not* Invoke § 112, ¶ 6,” yet under that heading, never points to any language in claim 21 in support of the proposition that “customization module” is not a means-plus-function term. Appellant’s Opening Br. at 18–27. Grecia complains that the district court ignored the claim language and specification, but Grecia’s Appeal Brief hardly addresses the customization module. Instead, Grecia only refers in passing to dependent claims 23 and 24. Grecia’s only argument on appeal appears to be that dependent claims 23 and 24 provide structure for claim 21.

But the limitations of claims 23 and 24 are not recited in claim 21, and nothing in these dependent claims demonstrates that the “customization module” is a term commonly understood by persons of skill to denote a specific algorithm or other structure. Moreover, each of these dependent claims attempts to further define the customization module by self-reference to its own “customization” function. See ’860 patent at claim 23 (“wherein the *customization* module *customizes* the tag”) (emphases added); *id.* at claim 24 (“wherein the *customization* module *customizes* the user access panel”) (emphases added). These limitations are purely functional and fail to describe any structure. Grecia appears to admit as much by concluding that

the customization module “equips the computer product with the means to manipulate the ‘verification token’ to meet the user’s purpose.” Appellant’s Opening Br. at 21.

In his Reply Brief, Grecia insists that claim 21 teaches *how* the customization module operates. But instead of citing any language in claim 21 that explains how customization is performed, Grecia points again to the purely functional limitations of dependent claims 23 and 24 and also to portions of the specification that disclose only the results of customization. *See* ’860 patent at col. 6, ll. 26–33 (“customization module 206 facilitates adding one or more of a banner, a logo, an image, an advertisement, a tag line, a header message and textual information to the user access panel of the encrypted digital media”); *id.* at Fig. 3 (depicting a graphical user interface 301 with a particular message for the user). The only language that Grecia identifies in claim 21 is the recitation of the “customization module” itself and the verification token as a type of permission with various representations. *See* ’965 patent at claim 21 (“wherein the at least one of purchase permission, rental permission, or membership permission being represented by one or more of a tag, a letter, a number, a combination of letters and numbers, a successful payment, a rights token, a phrase, a name, a membership credential, an image, a logo, a service name, an authorization, a list, an interface button, a downloadable program, or the redeemable instrument”). Contrary to Grecia’s assertions, nothing in claim 21 explains *how* the customization module operates. At bottom, nothing in the claims or specification suggests that “customization module” would have been understood by persons of skill in the art to have a sufficiently definite meaning as the name for structure.

Unlike in *Zeroclick, LLC v. Apple Inc.*, 891 F.3d 1003 (Fed. Cir. 2018), where we held that a person of skill would recognize the claimed “program” and “user interface code” as “specific references to conventional graphical user interface programs or code,” *id.* at 1008, the “customization

module” is a black box recitation untethered to any specific structure. Instead, like the “distributed learning control module” that we held to be a means-plus-function term in *Williamson*, the “customization module” is directed to the “module” nonce word, the prefix to the “module” word imparts no structure, and the specification also fails to impart any structural significance to the term. *See Williamson*, 792 F.3d at 1351.

Grecia’s reliance on *Apple Inc. v. Motorola, Inc.*, 757 F.3d 1286 (Fed. Cir. 2014) is also misplaced.<sup>2</sup> As an initial matter, our decision in *Apple* that the term “heuristic” should not be construed as a means-plus-function claim was under a strong presumption against the application of means-plus-function in the absence of the word “means.” *Id.* at 1304. But that strong presumption was overruled in *Williamson*. *Williamson*, 792 F.3d at 1349. Moreover, in *Apple*, we noted that the “heuristic” term at issue had a “known meaning” with an explanation of *how* to achieve the output of the claimed “heuristic” based on specific rules such as the initial angle of finger contact with the screen, the number of fingers making contact, and the direction of movement of finger contact. *Apple*, 757 F.3d at 1300, 1303. But, as we have explained, nothing in the ’860 patent or claims suggests that “customization module” has a known meaning or describes how the customization module operates.

---

<sup>2</sup> Grecia also misapplies our holding in *Med. Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, (Fed. Cir. 2003) in support of his argument that the customization module is not a means-plus-function term. In *Med. Instrumentation*, there was no dispute that the limitation at issue was written in means-plus-function form and therefore we did not need to address the threshold question of whether § 112, ¶ 6 applies. *Id.* at 1210.

Because the term “customization module” does not describe anything structural, the district court did not err in concluding that claim 21 is subject to 35 U.S.C. § 112, ¶ 6.

B. The Specification Does Not Disclose Sufficient Structure Corresponding to the Claimed Function

Because the term “customization module” is a means-plus-function term, we must construe the term “to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.” 35 U.S.C. § 112, ¶ 6. A disclosed structure is a “corresponding structure” only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim. *Williamson*, 792 F.3d at 1352. For a computer-implemented means-plus-function term, the corresponding structure is typically the algorithm disclosed in the specification for performing the claimed function. *Id.*; *Aristocrat Techs. Austl. Pty Ltd. v. Int’l Game Tech.*, 521 F.3d 1328, 1333 (Fed. Cir. 2008).

Here, the only claimed functionality of the term “customization module” in claim 21 arises from the prefix “customization.” But the specification fails to explain how such customization is performed. Instead, the specification only describes the results of customization, i.e., customizing a user access panel of encrypted digital media with information such as “a banner, a logo, an image, an advertisement, a tag line, a header message and textual information.” ’860 patent at col. 6, ll. 26–33; *see also id.* at claims 23–24. We have held that describing “the results of the operation of an unspecified algorithm” is not sufficient to transform the disclosure of a general-purpose computer into the disclosure of sufficient structure to satisfy § 112, ¶ 6. *Aristocrat Techs.*, 521 F.3d at 1335 (Fed. Cir. 2008). Because the ’860 specification merely describes the results of customization without any algorithm for configuring the claimed module to obtain those results, we agree with the district court that the specification fails to disclose the



“corresponding structure” required under § 112, ¶ 6, thus rendering claim 21 indefinite under § 112, ¶ 2.

#### CONCLUSION

We have considered Grecia’s remaining arguments and find them unpersuasive. For the reasons stated above, we *affirm* the district court’s conclusion that claim 21 is indefinite because the “customization module” term of claim 21 invokes § 112, ¶ 6 and the specification fails to disclose any structure corresponding to the claimed function. We therefore need not reach the district court’s conclusions with regards to the “metadata” and “computer product” terms. Grecia makes no separate arguments for the asserted dependent claims, so we affirm the district court’s invalidity ruling as to those claims as well.

#### AFFIRMED

# EXHIBIT 24

[Trials@uspto.gov](mailto:Trials@uspto.gov)

Tel: 571-272-7822

Paper 13

Entered: September 27, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

MASTERCARD INTERNATIONAL INCORPORATED,  
Petitioner,

v.

WILLIAM GRECIA,  
Patent Owner.

---

Cases IPR2017-00791  
Patent 8,533,860 B1

---

Before JAMESON LEE, MICHAEL W. KIM, and  
MICHELLE N. WORMMEESTER, *Administrative Patent Judges*.

KIM, *Administrative Patent Judge*.

JUDGMENT

*37 C.F.R. § 42.73(b)*

IPR2017-00791  
Patent 8,533,860 B1

On January 27, 2017, Mastercard International Incorporated, (“Petitioner”) filed a Petition (Paper 2) requesting *inter partes* review of claims 1–30 of U.S. Patent No. 8,533,860 B1 (Ex. 1001, “the ’860 patent”). On July 5, 2017, we instituted trial in a Decision with respect to claims 1–8 and 11–20 of the ’860 patent. Paper 7, 49.

On September 18, 2017, William Grecia (“Patent Owner”) contacted the Board to request authorization to file a Motion to Request Adverse Judgment (“Motion”) on all instituted claims 1–8 and 11–20. With the request, Patent Owner also included the instant Motion. The Board grants the request, and for the purposes of expediency, we enter *sua sponte* the included Motion into the record, and consider the Motion at this time.

Under 37 C.F.R. § 42.73(b)(2), a party may request judgment against itself at any time during a proceeding, and cancellation or disclaimer of a claim such that the party has no remaining claim in the trial is an action construed to be a request for entry of adverse judgment. In the circumstances of this case, the filing of a “motion” is not necessary. Patent Owner simply could have filed a request for adverse judgment. The Motion states “Patent Owner respectfully requests judgment against itself as to the claims remaining in this proceeding and asks that the Board cancel claims 1–8 and 11–20.” Motion 1–2. Accordingly, on these facts, we treat the Motion as a request for adverse judgment.

It is

ORDERED that Patent Owner’s Motion is *granted* and the Motion is entered into the record;

FURTHER ORDERED that the request is *granted* and adverse judgment is entered against Patent Owner;

IPR2017-00791

Patent 8,533,860 B1

FURTHER ORDERED that claims 1–8 and 11–20 of U.S. Patent No. 8,533,860 B1 be cancelled; and

FURTHER ORDERED that the *inter partes* review captioned IPR2017-00791 is terminated.

IPR2017-00791

Patent 8,533,860 B1

PETITIONER:

Joseph R. Lanser

Brian Michaelis

David A. Klein

Joseph Walker

SEYFARTH SHAW LLP

[jlanser@seyfarth.com](mailto:jlanser@seyfarth.com)

[bmichaelis@seyfarth.com](mailto:bmichaelis@seyfarth.com)

[daklein@seyfarth.com](mailto:daklein@seyfarth.com)

[jmwalker@seyfarth.com](mailto:jmwalker@seyfarth.com)

PATENT OWNER:

Isaac Rabicoff

RABICOFF LAW LLC

[isaac@rabilaw.com](mailto:isaac@rabilaw.com)